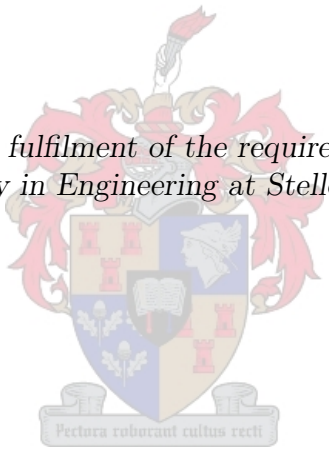
Conflict Detection and Resolution for Autonomous Vehicles

by

Corné Edwin van Daalen



Dissertation presented in fulfilment of the requirements for the degree of Doctor of Philosophy in Engineering at Stellenbosch University

Promoter: Professor Thomas Jones
Department of Electrical and Electronic Engineering

March 2010

Declaration

By submitting this dissertation electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the owner of the copyright thereof (unless to the extent explicitly otherwise stated) and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

March 2010

Abstract

Autonomous vehicles have recently received much attention from researchers. The prospect of safe and reliable autonomous vehicles for general, unregulated environments promises several advantages over human-controlled vehicles, including increased efficiency, reliability and capability with the associated decrease in danger to humans and reduction in operating costs. A critical requirement for the safe operation of fully autonomous vehicles is their ability to avoid *collisions* with obstacles and other vehicles. In addition, they are often required to maintain a minimum separation from obstacles and other vehicles, which is called *conflict* avoidance. The research presented in thesis focuses on methods for effective conflict avoidance.

Existing conflict avoidance methods either make limiting assumptions or cannot execute in real-time due to computational complexity. This thesis proposes methods for real-time conflict avoidance in uncertain, cluttered and dynamic environments. These methods fall into the category of non-cooperative conflict avoidance. They allow very general vehicle and environment models, with the only notable assumption being that the position and velocity states of the vehicle and obstacles have a jointly Gaussian probability distribution.

Conflict avoidance for fully autonomous vehicles consists of three functions, namely modelling and identification of the environment, conflict detection and conflict resolution. We present an architecture for such a system that ensures stable operation.

The first part of this thesis comprises the development of a novel and efficient probabilistic conflict detection method. This method processes the predicted vehicle and environment states to compute the probability of conflict for the prediction period. During the method derivation, we introduce the concept of the flow of probability through the boundary of the conflict region, which enables us to significantly reduce the complexity of the problem. The method also assumes Gaussian distributed states and defines a tight upper bound to the conflict probability, both of which further reduce the problem complexity, and then uses adaptive numerical integration for efficient evaluation. We present the results of two simulation examples which show that the proposed method can calculate in real-time the probability of conflict for complex and cluttered environments and complex vehicle maneuvers, offering a significant improvement over existing methods.

The second part of this thesis adapts existing kinodynamic motion planning algorithms for conflict resolution in uncertain, dynamic and cluttered environments. We use probabilistic roadmap methods and suggest three changes to them, namely using probabilistic conflict detection methods, sampling the state-time space instead of the state space and batch generation of samples. In addition, we propose a robust and adaptive way to choose the size of the sampling space using a maximum least connection cost bound. We then put all these changes together in a proposed motion planner for conflict resolution. We present the results of two simulation examples which show that the proposed motion planner can only find a feasible path in real-time for simple and uncluttered environments. However, the manner in which we handle uncertainty and the sampling space bounds offer significant contributions to the conflict resolution field.

Opsomming

Outonome voertuie het die afgelope tyd heelwat aandag van navorsers geniet. Die vooruitsig van veilige en betroubare outonome voertuie vir algemene en ongereguleerde omgewings beloof verskeie voordele bo menslik-beheerde voertuie en sluit hoër effektiwiteit, betroubaarheid en vermoëns asook die gepaardgaande veiligheid vir mense en laer bedryfskoste in. 'n Belangrike vereiste vir die veilige bedryf van volledig outonome voertuie is hul vermoë om *botsings* met hindernisse en ander voertuie te vermy. Daar word ook dikwels van hulle vereis om 'n minimum skeidingsafstand tussen hulle en die hindernisse of ander voertuie te handhaaf – dit word *konflik*vermyding genoem. Die navorsing in hierdie tesis fokus op metodes vir effektiewe konflikvermyding.

Bestaande konflikvermydingsmetodes maak óf beperkende aannames óf voer te stadig uit as gevolg van bewerkingskompleksiteit. Hierdie tesis stel metodes voor vir intydse konflikvermyding in onsekere en dinamiese omgewings wat ook baie hindernisse bevat. Die voorgestelde metodes val in die klas van nie-samewerkende konflikvermydingsmetodes. Hulle kan algemene voertuig- en omgewingsmodelle hanteer en hul enigste noemenswaardige aanname is dat die posisie- en snelheidstoestande van die voertuig en hindernisse Gaussiese waarskynliksheidverspreidings toon.

Konflikvermyding vir volledig outonome voertuie bestaan uit drie stappe, naamlik modellering en identifikasie van die omgewing, konflikdeteksie en konflikresolusie. Ons bied 'n argitektuur vir so 'n stelsel aan wat stabiele werking verseker.

Die eerste deel van die tesis beskryf die ontwikkeling van 'n oorspronklike en doeltreffende metode vir waarskynliksheid-konflikdeteksie. Die metode gebruik die voorspelde toestande van die voertuig en omgewing en bereken die waarskynlikheid van konflik vir die betrokke voorspellingsperiode. In die afleiding van die metode definiëer ons die konsep van waarskynliksheidvloei oor die grens van die konflikdomein. Dit stel ons in staat om die kompleksiteit van die probleem beduidend te verminder. Die metode aanvaar ook Gaussiese waarskynlikheidsverspreiding van toestande en definiëer 'n nou bogrens tot die waarskynlikheid van konflik om die kompleksiteit van die probleem verder te verminder. Laastens gebruik die metode aanpasbare integrasiemetodes vir vinnige berekening van die waarskynlikheid van konflik. Die eerste deel van die tesis sluit af met twee simulasies wat aantoon dat die voorgestelde konflikdeteksiemetode in staat is om die waarskynlikheid van konflik intyds te bereken, selfs vir komplekse omgewings en voertuigbewegings. Die metode lewer dus 'n beduidende bydrae tot die veld van konflikdeteksie.

Die tweede deel van die tesis pas bestaande kinodinamiese beplanningsalgoritmes aan vir konflikresolusie in komplekse omgewings. Ons stel drie veranderings voor, naamlik die gebruik van waarskynliksheid-konflikdeteksiemetodes, die byvoeg van 'n tyd-dimensie in die monsterruimte en die generasie van meervoudige monsters. Ons stel ook 'n robuuste en aanpasbare manier voor om die grootte van die monsterruimte te kies. Al die voorafgaande voorstelle word saamgevoeg in 'n beplanner vir konflikresolusie. Die tweede deel van die tesis sluit af met twee simulasies wat aantoon dat die voorgestelde beplanner slegs intyds 'n oplossing kan vind vir eenvoudige omgewings. Die manier hoe die beplanner onsekerheid hanteer en die begrensing van die monsterruimte lewer egter waardevolle bydraes tot die veld van konflikresolusie.

Acknowledgements

I would like to thank the following people for their contribution towards this project.

- Professor Thomas Jones for his support and guidance, as well as those ideas that have proved so successful in this thesis.
- The Institute for Maritime Technology in Simons Town, South Africa for their financial support for the research as well their interest and hospitality during feedback sessions.
- Keith Browne, Riaan Doorduin, Nicol Carstens, Izak Marais and AM de Jager for reviewing sections of this thesis and the paper on which part of it is based.
- Izak, Steven, Arno and the rest of the people in the Electronic Systems Laboratory at the University of Stellenbosch for a pleasant two years.
- My wife, Liesbet, for her love, support and patience during my studies.

Contents

Nomenclature

Acronyms

AI	Artificial Intelligence
ATC	Air Traffic Control
AUV	Autonomous Underwater Vehicle
BVP	Boundary Value Problem
DATMO	Detection and Tracking of Moving Obstacles
EKF	Extended Kalman Filter
GPWS	Ground Proximity Warning System
GPS	Global Positioning System
IMU	Inertial Measurement Unit
LPM	Local Planning Method
PID	Proportional-integral-derivative
PRM	Probabilistic Roadmap
ROV	Remotely Operated Vehicle
RPP	Randomised Potential Field Planner
RRT	Rapidly-exploring Random Tree
SISO	Single-input single-output
SLAM	Simultaneous Localisation and Mapping
SOC	System Operating Characteristic
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
UKF	Unscented Kalman Filter
USV	Unmanned Surface Vehicle

Symbol Conventions

x	Scalar
$\mathbf{x}$	Vector
$\mathbf{x}(t)$	Time-varying vector
X	Set
$\mathbf{X}$	Matrix
$\mathbf{X}(t)$	Time-varying matrix
$\mathbf{X}(t, \omega)$	Vector of random processes
$\dot{x}$	Derivative of x
$\overline{x}$	Mean of x

List of Symbols

A_t	Set of outcomes for which a conflict condition exists during $[t_0, t]$
$A_{t:t+\Delta t}$	Set of outcomes for which a conflict condition exists during $[t, t+\Delta t]$
$\mathcal{A}_{\boldsymbol{\mu}}$	Set of all admissible inputs
B_t	Set of outcomes for which no conflict condition exists during $[t_0, t]$
$\mathbf{C}_{\mathbf{R}}(t,t)$	Covariance matrix associated with $\mathbf{R}(t,\omega)$
C_t	Conflict region indexed at time t
$\mathbf{C}_{\mathbf{V}}(t,t)$	Covariance matrix associated with $\mathbf{V}(t,\omega)$
$\mathbf{C}_{\mathbf{X}}(t,t)$	Covariance matrix associated with $\mathbf{X}(t,\omega)$
$C\left(C_t\right)$	Boundary curve of conflict area C_t
$D_t^{\mathbf{p}}$	Set of outcomes for which $\mathbf{R}(t,\omega) = \mathbf{p}$
ΔS	Rectangular surface element on conflict volume boundary
ΔV	Rectangular column on ΔS
Ef	Error rule for integrand f
$f_{\mathbf{R}(t)}(\mathbf{p})$	Probability density function associated with $\mathbf{R}(t,\omega)$
$f_{\mathbf{R}(t)}^{B_t}(\mathbf{p})$	Conditional probability density function $f_{\mathbf{R}(t)}(\mathbf{p}\|B_t)$
$f_{V_n}(v_n\|D_t^{\mathbf{p}})$	Probability density function of $V_n(t,\omega)$ given $\mathbf{R}(t,\omega) = \mathbf{p}$
$f_{V_n}^{B_t}(v_n\|D_t^{\mathbf{p}})$	Conditional probability density function $f_{V_n}(v_n\|D_t^{\mathbf{p}} \cap B_t)$
$\mathcal{F}_{\boldsymbol{\mu}}$	Set of all feasible inputs
$\mathcal{G}_T$	Directed graph containing the set of known reachable points and connecting feasible inputs
$J(\mathbf{y}, \boldsymbol{\mu})$	Cost function associated with trajectory $\mathbf{y}(t)$ and input $\boldsymbol{\mu}(t)$
J_B	Maximum least connection cost bound
$\mathcal{M}$	Maneuver space
$\mathbf{n}$	Unit vector normal to the conflict volume surface
$N(\mathbf{a}, \mathbf{B})$	Normal distribution with mean $\mathbf{a}$ and covariance $\mathbf{B}$
$\mathbf{p}$	Position variable
$P\big[Q\big]$	Probability of event Q
$P_C(t)$	Probability of conflict for time period $[t_0, t]$
$P_C^*(t)$	Lower bound to the probability of conflict for time period $[t_0, t]$
$P_C^{\mathrm{UB}}(t)$	Upper bound to the probability of conflict for time period $[t_0, t]$
P_C^{MAX}	Conflict probability threshold for safe operation
$\varphi_{\boldsymbol{\mu}}(\mathbf{y}_0, t)$	Vehicle trajectory induced by input $\boldsymbol{\mu}(t)$ for initial states $\mathbf{y}_0$
Qf	Integration rule for integrand f
Q_M	Set of all maneuvers of a maneuver automaton
Q_T	Set of all trim trajectories of a maneuver automaton
$\mathbf{R}(t,\omega)$	Vehicle position states
$\mathcal{R}(\mathbf{y}, t)$	Reachable set for the pair $(\mathbf{y}, t)$
$S\left(C_t\right)$	Surface of conflict volume C_t
σ	Standard deviation
t	Time variable
$\mathcal{T}$	Time domain

t_f	Final time instant
t_0	Initial time instant
$\mathbf{u}(t)$	Input to state space model
$\mathcal{U}$	Input space of state space model
$\boldsymbol{\mu}(t)$	Input to general system model
$\mathbf{V}(t,\omega)$	Vehicle velocity states
$\mathcal{V}$	Input space of general model
$V_n(t,\omega)$	The component of $\mathbf{V}(t,\omega)$ normal to $\mathbf{n}$
$\mathbf{W}(t,\omega)$	Noise input
ω	Outcome variable
Ω	Set of all possible outcomes
$\mathbf{X}(t,\omega)$	Continuous states of state space model
$\mathcal{X}$	Continuous vehicle state space
$\mathbf{Y}(t,\omega)$	States of general model
$\mathcal{Y}_F$	Set of goal state-time pairs

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In the past two decades we have seen a marked increase in research into autonomous vehicles – that is, vehicles requiring no human intervention to operate. Autonomous systems such as the competitors in the DARPA Urban Challenge [5, 75] and military unmanned aerial vehicles (UAVs) such as the Global Hawk [9] have become well-known. However, despite these successes, the existing systems are either experimental or only used in regulated environments and there remains much work to be done before autonomous vehicles can effectively operate in unregulated environments.

A critical requirement for the safe operation of autonomous vehicles is their ability to avoid conflict with obstacles or other vehicles, which includes avoiding collisions. This ability to avoid conflict emanates from the conflict avoidance system on the autonomous vehicle and this thesis focuses on the methods used in this system.

Researchers have investigated the use of automated conflict avoidance systems for a wide range of applications, including air traffic control (ATC) [80], autonomous underwater vehicles (AUVs) [19], unmanned aerial vehicles (UAVs) [65], cars [36], ships [73], unmanned surface vehicles (USVs) [46] and satellites [63].

Conflict avoidance proved to be problematic and despite much effort from researchers, the existing methods do not entire succeed in providing a timely and reliable response. The difficulties are partly caused by the inherent system uncertainty, which includes sensor noise, uncertainty in obstacle identification, vehicle actuator uncertainty, modelling errors, unknown pilot or driver intent as well as uncertainty in the future obstacle trajectories and environment states. Many of the methods used in conflict avoidance are also computationally expensive. It is important for the conflict avoidance system to react quickly to sudden conflict situations and the methods should therefore execute in real-time. As a result, research into conflict avoidance focuses on designing efficient methods that are robust with respect to system uncertainty.

1.2 Definition of Conflict

This thesis is concerned with *conflict* avoidance. Conflict avoidance differs from *collision* avoidance – where the autonomous vehicle should not collide with other vehicles or obstacles – in that it requires the autonomous vehicle to maintain a minimum separation between itself and other vehicles or obstacles. *Conflict* is therefore defined as the situation where another vehicle or obstacle intrudes into an exclusion zone surrounding the autonomous vehicle. This exclusion zone is called the conflict region.

The size and shape of the conflict region are choices in the design of the conflict avoidance system. The choice of the conflict region size is a trade-off between safety and functionality –

a large conflict region will produce a safer system, but the system might be overly sensitive to the presence of obstacles and its maneuverability might be limited near obstacles.

The definition of conflict as stated above places constraints on the position states of the autonomous vehicle. An extension of this concept is the general state avoidance where the constraints due to conflict with other vehicles and obstacles are combined with constraints on the full state space of the autonomous vehicle, for instance to ensure that the vehicle stays within its performance envelope. However, in this study we only use the definition of conflict that places constraints on the autonomous vehicle position states.

1.3 Architecture of a Conflict Avoidance System

The intended application of the methods presented herein is to fully autonomous vehicles that could function in dynamic, cluttered and uncertain environments. We envisage that the conflict avoidance system on a fully autonomous vehicle would consist of three integral modules, namely a modelling module, a conflict detection module and a conflict resolution module. Such a conflict avoidance system with its inputs, outputs and internal data flows is shown in Figure 1.1. We

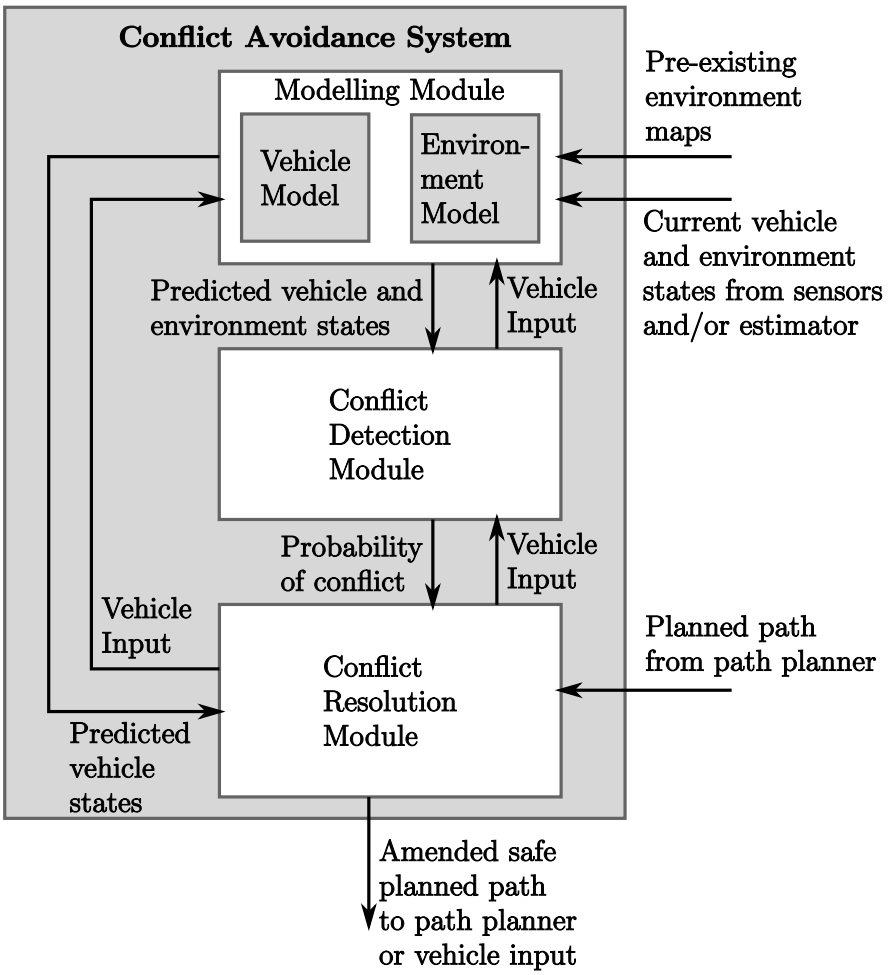


Figure 1.1: Conflict avoidance system architecture

adapted the diagram from Figure 2.1 in the thesis by Jones [37]. This type of conflict avoidance system architecture is no new concept, but the manner in which the conflict resolution module

interacts with the conflict detection module and the path planner is an approach that we have not encountered previously. A discussion of the modules and their interfaces follows below.

The *modelling module* contains the vehicle model and environment model. The vehicle model takes the current vehicle state estimate and the planned vehicle input and then generates the predicted vehicle states for the finite-length prediction period in question. This vehicle state propagation is used by both the conflict detection module and the conflict resolution module. The environment model fuses information from pre-existing environment maps with current sensor information, it identifies and track moving obstacles and provides a prediction of the future environment states for the prediction period in question. This environment state prediction is then passed to the conflict detection module when requested.

The *conflict detection module* determines the probability of conflict for a planned vehicle path segment. The vehicle path is induced by the vehicle input and is calculated by a simulation of the vehicle model for the given input. This simulation is performed by the modelling module which passes the predicted vehicle and environment states back to the conflict detection module. The conflict detection module then uses these predicted states to calculate the probability of conflict which is passed to the conflict resolution module.

The *conflict resolution module* attempts to find a vehicle input that would induce a path segment with a probability of conflict below a specified threshold. In addition, the chosen input should steer the vehicle towards the next waypoint or goal state. The input to the path planner is the next segment of the planned path to be executed as generated by the path planner. Before the vehicle is commanded to follow this path segment, the path segment is passed to the conflict resolution module, which passes it to the conflict detection module. If the conflict probability associated with this path segment is higher than the threshold, the conflict resolution module attempts to find an alternative path to the next waypoint on the planned path while keeping the conflict probability below the threshold. In addition to generating a path with low conflict probability, the conflict resolution module also attempts to find the optimal path with respect to some cost function. If a safe path to the next waypoint is found, it is passed back to the path planner or passed directly to the vehicle input. If no safe path to the next waypoint is found, a safe path optimised according to some criterion, such as the degree of exploration of unknown region, is returned.

The main focus of this thesis is on the methods used in the conflict detection module – it is the subject of Part I, where a computationally efficient algorithm to calculate the probability of conflict is developed. The methods used in the conflict resolution module is the subject of Part II, where we investigate how well the algorithm developed in Part I integrate and perform with existing conflict resolution methods. The vehicle models used in the modelling module is the subject of Chapter 2, while the environment models are discussed in Section 3.2. A short discussion of the methods used to identify and update the environment model is found in Section 2.4.

1.4 Research Objectives

This thesis presents a novel and efficient method for probabilistic conflict detection, as well as the adaptation of existing motion planning methods for probabilistic conflict resolution. We address the non-cooperative conflict avoidance problem for uncertain, cluttered and dynamic environments. We make only one notable assumption, namely that the probability distribution of the vehicle and environment position and velocity states is jointly Gaussian. Apart from that, the methods are kept general.

The focus of the research presented in this thesis is on finding computationally efficient methods for conflict detection and resolution that are able to execute in real-time for systems with uncertainty. The research objectives can be summarised in the following three points:

1. The main objective of the research is designing a computationally efficient method to calculate the probability of conflict for normally distributed vehicle and environment states. The method should execute in real-time even for cluttered and complex environments and complex[1] vehicle maneuvers.

2. A secondary objective is adapting existing conflict resolution methods for uncertain, dynamic and cluttered environments. The adapted method should also be efficient and should calculate a safe alternative vehicle path in real-time when conflict is predicted along the initial planned path.

3. Another secondary objective is to provide a unified conflict avoidance architecture for fully autonomous vehicles. The interaction between the different modules in the conflict avoidance system and between the conflict avoidance system and other vehicle systems should be stable.

The funding for the research presented in this thesis was provided with the application of autonomous underwater vehicles (AUVs) in mind. However, we present methods that are applicable to many types of autonomous vehicles, including AUVs. We motivate the applicability of the methods in this thesis to AUVs, but we do not restrict their application to AUVs.

1.5 Overview of Thesis

Chapter 1, the first of the opening chapters, introduces the research with a short background of autonomous vehicles and conflict avoidance. Thereafter it defines the concept of conflict, proposes an architecture for the conflict avoidance system of a fully autonomous vehicle, details the research objectives and concludes with an overview of the thesis.

The second of the opening chapters, Chapter 2, discusses the vehicle models to which the methods of this thesis can be applied. It deals with the state space and maneuver automaton representations before defining a general model description. This description includes the preceding two representations and is introduced to simplify notation. The chapter then supplies a synopsis of autonomous underwater vehicles (AUVs) and argues that the preceding vehicle models – and therefore also the methods presented in this thesis – can be applied to them.

Part I. The first part of the main thesis body presents the development of an efficient probabilistic conflict detection method forming the core contribution of this thesis.

Chapter 3 outlines conflict detection. It initially defines the three different approaches to conflict detection, of which we choose the probabilistic approach. This is followed by a definition of the probabilistic conflict detection problem and a discussion of important conflict detection concepts. Thereafter, the chapter concludes with an overview of existing probabilistic conflict detection methods.

The derivation of the novel conflict detection method, using the concept of probability flow is detailed in Chapter 4. It starts by defining probability flow and then derives an expression for the probability flow through the conflict region boundary for the general case. Thereafter, it makes two simplifications, namely calculating a tight upper bound to the probability of conflict and assuming that the vehicle position and velocity states have a jointly Gaussian probability distribution. Lastly, it presents an adaptive integration method used to numerically evaluate the integrals in the expression for the conflict probability.

Chapter 5 provides two example implementations to illustrate the real-time performance of the probability flow method: the first, two-dimensional example consists of two airplanes with crossing flight paths, while the second, three-dimensional example consists of an AUV in a cluttered harbour environment.

[1] As opposed to simple trajectories such as straight lines.

Part II. The second part of the main thesis body looks at adapting existing kinodynamic motion planning methods for use in conflict resolution.

Chapter 6 reviews conflict resolution. It starts by detailing the two different approaches to conflict resolution, of which we choose the replanning approach. Next, it discusses important planning concepts, formally defines the conflict resolution problem and surveys existing kinodynamic planning methods that could be used for conflict resolution.

Chapter 7 presents a motion planner for conflict resolution. It describes problems that existing methods encounter in uncertain, cluttered and dynamic environments and proposes some changes to the existing methods in order to handle these environments. Next, it suggests a manner in which to bound the sampling region, using the maximum least connection cost criterion. It then combines all the preceding proposed changes and presents a motion planning algorithm for conflict resolution.

Chapter 8 takes the example simulations of Chapter 5 and applies the motion planner proposed in Chapter 7 for conflict resolution.

The closing chapter, Chapter 9, briefly summarises the results presented in this thesis. It then lists the primary contributions and discusses promising avenues to further research.

Chapter 2

Modelling Autonomous Vehicles

The conflict detection and resolution methods presented in this thesis are *model predictive* techniques. This means that the vehicle and environment states are propagated into the future based on dynamic models. Conflict detection methods process these propagated states to predict the occurrence of any conflict. Should conflict be predicted, the conflict resolution methods use the vehicle model to design alternative vehicle commands in order to avoid future conflict.

In this chapter, we discuss the vehicle models that are compatible with the methods presented in this thesis. Firstly, we explore the commonly-used state space representation (Section 2.1), thereafter we briefly discuss an alternative model – the maneuver automaton representation (Section 2.2). Section 2.3 presents a unified notation for the preceding representations. The chapter concludes in Section 2.4 with an overview of autonomous underwater vehicles and a discussion of their compatibility with the models of the preceding sections. The environment model that is compatible with the methods in this thesis is discussed separately in Chapter 3.

2.1 State Space Representation

The standard manner in which the dynamics of a vehicle are represented, is a state space model. This section therefore covers the type of state space model to which the methods in this thesis can be applied. Subsection 2.1.1 outlines the general state space formulation, followed by a detailed description of the time-variant linear state space model, which is a popular model used for autonomous vehicles, in Subsection 2.1.2.

2.1.1 General Definition

The state space representation is given in general form as [25, 48]

$$\dot{\mathbf{x}}(t) = f\left(\mathbf{x}(t), \mathbf{u}(t)\right), \tag{2.1}$$

where $\mathbf{x}(t) \in \mathcal{X}$ denotes the vehicle states with $\mathcal{X}$ the state space, which is an n-dimensional smooth manifold, and $\mathbf{u}(t) \in \mathcal{U}$ is the input with $\mathcal{U} \subseteq \mathbb{R}^m$ the input space. The definitions of the state and input spaces combines with Equation 2.1 place constraints on the propagation of the vehicle states. Additional inequality constraints are sometimes placed on the vehicle input and states, given by [26]

$$F\left(\mathbf{x}(t), \mathbf{u}(t)\right) \leq \mathbf{0}, \tag{2.2}$$

where

$$F\left(\mathbf{x}(t), \mathbf{u}(t)\right) \triangleq \begin{bmatrix} F_1\left(\mathbf{x}(t), \mathbf{u}(t)\right) \\ F_2\left(\mathbf{x}(t), \mathbf{u}(t)\right) \\ \vdots \\ F_N\left(\mathbf{x}(t), \mathbf{u}(t)\right) \end{bmatrix} \tag{2.3}$$

and the inequality operator in Equation 2.2 is applied element-wise. All the above constraints on the propagation of the vehicle state are called *differential constraints*.

To ensure the safe operation of autonomous vehicles, it is required that they avoid conflict with obstacles or other vehicles according to the definition of conflict in Section 1.2. This requirement induces another set of constraints, namely *global constraints*, on the vehicle. Formal definitions of obstacles, conflict and global constraints are explained in Section 3.2.

Uncertainty in the vehicle sensors and actuators as well as in the environment causes uncertainty in the current vehicle states. This current state uncertainty as well as uncertainty in the future environment states (such as uncertain wind velocity predictions, unknown pilot intent and uncertain future obstacle positions) and uncertainty in the future vehicle states due to disturbances, inaccurate models and sensing errors cause uncertainty in the future vehicle states relative to the environment. In many cases it is necessary to model this uncertainty, therefore, Equation 2.1 is amended to read

$$\dot{\mathbf{X}}(t,\omega) = f\left(\mathbf{X}(t,\omega), \mathbf{u}(t), \mathbf{W}(t,\omega)\right), \tag{2.4}$$

where $\mathbf{W}(t,\omega)$ is a vector of random processes with outcome $\omega \in \Omega$, and the vehicle state, $\mathbf{X}(t,\omega)$, and its derivative, $\dot{\mathbf{X}}(t,\omega)$, are now also random processes. The inequality constraint of Equation 2.2 is now only applied to the mean state values.

This thesis is applicable to all systems of which the joint distribution of the position states $\mathbf{R}(t,\omega) \in \mathbf{X}(t,\omega)$ and velocity states $\mathbf{V}(t,\omega) \in \mathbf{X}(t,\omega)$ is Gaussian or can be sufficiently accurately approximated as Gaussian, that is

$$\begin{bmatrix} \mathbf{V}(t,\omega) \\ \mathbf{R}(t,\omega) \end{bmatrix} \sim N\left(\begin{bmatrix} \overline{\mathbf{V}}(t) \\ \overline{\mathbf{R}}(t) \end{bmatrix}, \begin{bmatrix} \mathbf{C_V}(t,t) & \mathbf{C_{VR}}(t,t) \\ \mathbf{C_{VR}}^T(t,t) & \mathbf{C_R}(t,t) \end{bmatrix} \right). \tag{2.5}$$

This Gaussian state distribution assumption is reasonable because the vehicle and environment states are usually estimated using Kalman Filters (or derivatives thereof such as the Extended Kalman Filter or the Unscented Kalman Filter), which assume or approximate the state probability distribution as Gaussian. As an example, this assumption is also motivated for autonomous underwater vehicles in Section 2.4. For many autonomous vehicle systems, the probability distribution of the vehicle and environment position and velocity states can be approximated as Gaussian with sufficient accuracy. However, we recognise that this might not be the case for some systems, including vehicles with highly non-linear dynamics, non-linear controllers (such as "tight" cross-track control with "loose" along-track control), as well as some systems with discrete states.

The state space definition of Equation 2.4 could depict an open-loop or a closed-loop system. The use of a closed-loop system means that the stabilisation and tracking functions are incorporated in the model, which reduces the complexity of the conflict resolution and path planning systems. However, a closed-loop system typically contains a controller and estimator, the addition of which adds several dimensions to the vehicle state vector, therefore increasing the problem complexity.

It must be emphasised that although the focus of this research is on autonomous vehicles, the methods presented herein are applicable to all systems that conform to the descriptions in Equations 2.4 and 2.5.

This approach for conflict avoidance can be described as a model predictive approach: the vehicle and environment models are used to propagate the vehicle states relative to the environment. Following conflict prediction, the planned vehicle path is adjusted to generate predicted vehicle states without conflict. The techniques presented herein therefore rely heavily on the existence of computationally efficient methods to propagate the vehicle states. Propagation of the probability distribution in Equation 2.5 involves solving Equation 2.4 for an initial distribution and a specified time t. Although there is no solution to the general problem, there exists an efficient procedure to propagate the vehicle state distribution for time-variant linear models. This procedure is discussed in the next subsection.

2.1.2 Linear Definition

Many autonomous vehicles can be modelled or sufficiently accurately approximated by a time-variant linear system. When this is the case and the noise input $\mathbf{W}(t,\omega)$ is normally distributed, the process of propagating the mean and covariance of the vehicle state is relatively straightforward. We derive the equations below using a similar process to that used by Jones[37].

The time-variant linear state space system is given by

$$\dot{\mathbf{X}}(t,\omega) = \mathbf{A}(t)\mathbf{X}(t,\omega) + \mathbf{B}(t)\mathbf{u}(t) + \mathbf{B}_w(t)\mathbf{W}(t,\omega). \tag{2.6}$$

The noise $\mathbf{W}(t,\omega)$ is assumed to be white, zero-mean and Gaussian with a covariance matrix $\mathbf{Q}(t)$. In order to solve for $\mathbf{X}(t,\omega)$ in Equation 2.6, we require the integration of the white noise $\mathbf{W}(t,\omega)$. As shown by Borrie [11], white noise is not integrable in the Riemann sense. We therefore use the model

$$\mathrm{d}\mathbf{X}(t,\omega) = \mathbf{A}(t)\mathbf{X}(t,\omega)\,\mathrm{d}t + \mathbf{B}(t)\mathbf{u}(t)\,\mathrm{d}t + \mathbf{B}_w(t)\,\mathrm{d}\boldsymbol{\beta}(t,\omega), \tag{2.7}$$

where $\boldsymbol{\beta}(t,\omega)$ is a vector of extended Wiener processes with diffusion $\mathbf{Q}(t)$. From Borrie [11] and Grimble and Johnson [30], we state Equations 2.8 through 2.12. The solution of the vehicle states is stated as

$$\mathbf{X}(t,\omega) = \boldsymbol{\Phi}(t,t_0)\mathbf{X}(t_0,\omega) + \int_{t_0}^{t} \boldsymbol{\Phi}(\xi,t_0)\mathbf{B}(\xi)\mathbf{u}(\xi)\,\mathrm{d}\xi + \int_{t_0}^{t} \mathbf{B}_w(\xi)\,\mathrm{d}\boldsymbol{\beta}(\xi,\omega), \tag{2.8}$$

where the state transition matrix $\boldsymbol{\Phi}(t,t_0)$ is given by the solution of

$$\frac{d}{dt}\boldsymbol{\Phi}(t,t_0) = \mathbf{A}(t)\boldsymbol{\Phi}(t,t_0). \tag{2.9}$$

The propagation of the mean states is stated as

$$\dot{\overline{\mathbf{X}}}(t) = \mathbf{A}(t)\overline{\mathbf{X}}(t) + \mathbf{B}(t)\mathbf{u}(t), \tag{2.10}$$

where $\overline{\mathbf{X}}(t) \triangleq \mathcal{E}[\mathbf{X}(t,\omega)]$. The covariance of $\mathbf{X}(t,\omega)$ is given by

$$\begin{aligned}\mathbf{C}_{\mathbf{X}}(t,t) = {} & \boldsymbol{\Phi}(t,t_0)\mathbf{C}_{\mathbf{X}}(t_0,t_0)\boldsymbol{\Phi}^T(t,t_0) \\ & + \int_{t_0}^{t} \boldsymbol{\Phi}(\xi,t_0)\mathbf{B}_w(\xi)\mathbf{Q}(\xi)\mathbf{B}_w^T(\xi)\boldsymbol{\Phi}^T(\xi,t_0)\,\mathrm{d}\xi,\end{aligned} \tag{2.11}$$

where $\mathbf{C}_{\mathbf{X}}(t_1,t_2) \triangleq \mathcal{E}\left[\left(\mathbf{X}(t_1,\omega) - \overline{\mathbf{X}}(t_1)\right)\left(\mathbf{X}(t_2,\omega) - \overline{\mathbf{X}}(t_2)\right)^T\right]$. The propagation of the covariance of $\mathbf{X}(t,\omega)$ is then given by

$$\dot{\mathbf{C}}_{\mathbf{X}}(t,t) = \mathbf{A}(t)\mathbf{C}_{\mathbf{X}}(t,t) + \mathbf{C}_{\mathbf{X}}(t,t)\mathbf{A}^T(t) + \mathbf{B}_w(t)\mathbf{Q}(t)\mathbf{B}_w^T(t). \tag{2.12}$$

For the conflict detection method developed in Part I, we require the predicted mean and covariance of the vehicle state to be available at any time instant in the prediction period. For vehicles that can adequately be described by the time-variant linear model of Equation 2.7, the propagation of the mean and covariance of the vehicle state is given by Equations 2.10 and 2.12 respectively. These ordinary differential equations can then be solved by using a numerical method such as the Runge-Kutta method.

2.2 Maneuver Automaton Representation

The state space representation sometimes requires an excessive amount of information to define the system input over a time period. We therefore supply an alternative system representation – the maneuver automaton representation – that requires much less information to define the system input. This representation is especially useful to reduce the complexity of the conflict resolution algorithms.

Classically, the manner in which a controller in a state space representation is implemented, is to choose a fixed sampling period and then discretise the controller. A series of input values, each held constant over one sampling period, constitutes the input to the system over a certain time period. When designing an input signal, each of these elements in the input series can be considered as a variable to which a value has to be assigned. Combined with the fact that the input to the system is typically a vector consisting of several elements, discretising the controller causes the number of these variables to become excessive. In addition, both the input and induced vehicle states have to comply with the differential constraints on the system.

One way to simplify the construction of an input signal is to define a number of motion primitives [48]. Following the terminology of Frazzoli et al. [24], we define two types: trim trajectories and maneuvers. Trim trajectories are steady motions which can be executed for any length of time – called the coasting time – whereas maneuvers are transitions between different trim trajectories which execute in fixed time periods. The system can now be modelled as a hybrid system called a *maneuver automaton* where the continuous vehicle states as given in Subsection 2.1 are augmented with a discrete state, namely the motion primitives.

We now illustrate some of the features of a maneuver automaton by using the example of an autonomous helicopter. A graph of the example system (adapted from Figure 14.8 in the book by LaValle [48]) is shown in Figure 2.1. The set of trim trajectories of this au-

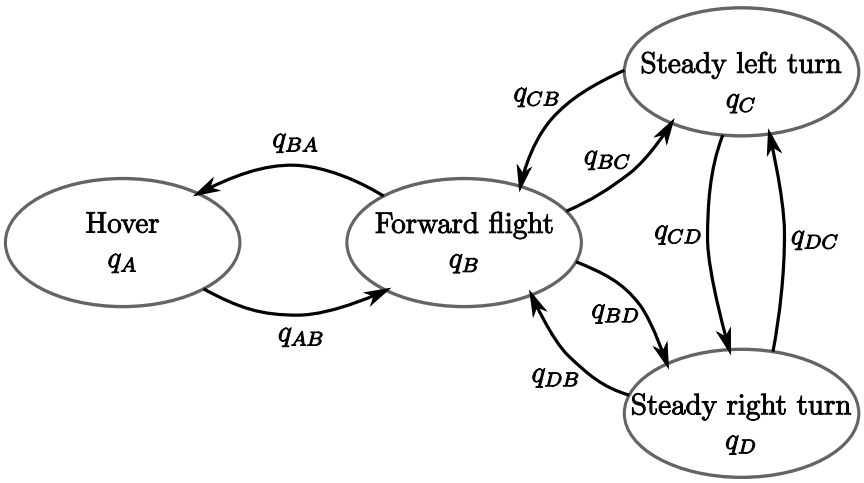


Figure 2.1: Maneuver automaton example

tonomous helicopter is given by $Q_T = \{q_A, q_B, q_C, q_D\}$ and the set of maneuvers is given by $Q_M = \{q_{AB}, q_{BA}, q_{BC}, q_{CB}, q_{BD}, q_{DB}, q_{CD}, q_{DC}\}$. The transition between a trim trajectory and a maneuver is a controlled jump and initiated by a command from a discrete input signal. The transition between a maneuver and a trim trajectory is an autonomous jump and occurs when the fixed-length maneuver execution time has elapsed. The continuous dynamics for each motion primitive is described by a state space model – each motion primitive could have a different continuous controller, different continuous input variables and therefore a different continuous closed-loop model. The input to the state space system is generated by a procedure specific

to each discrete state. When all the above-mentioned components are combined, the input to the maneuver automaton then is the ordered pair consisting of the coasting time of the current trim trajectory as well as the next maneuver to jump to. The input over a certain time period can be written as a series of ordered pairs, or $\{(\tau_1, q_1), (\tau_2, q_2), \ldots, (\tau_n, q_n)\}$ with $q_i \in Q_M$, τ_i the coasting time of the corresponding trim trajectory and $i \in \{k \in \mathbb{N} : k \leq n\}$. The number of variables in the input to a maneuver automaton is typically much less than the number of variables in the input to a discretised state space system.

To sum up the preceding discussion, the advantages of using the maneuver automaton representation are the following:

1. The number of variables necessary to describe an input signal is much less than that of a discretised state space system, dramatically reducing the complexity of designing an input signal.

2. The differential constraints on the vehicle state and input signal are incorporated in the motion primitive definitions. This removes the necessity of checking for these constraints when constructing an input signal. This in turn reduces the complexity of input signal design.

3. The mean and covariance of the vehicle position and velocity states could be propagated and stored beforehand for each motion primitive, or the propagation procedure could be customised and simplified for each motion primitive. These measures would remove the need to solve Equations 2.10 and 2.12 online.

The disadvantage of using the maneuver automaton representation is:

1. The set of mean vehicle states reachable by a maneuver automaton is a subset of the mean vehicle states reachable by the state space representation. It is therefore difficult to design the motion primitives such that the full dynamic capability of the vehicle is captured while keeping the number of motion primitives to a minimum.

The maneuver automaton representation therefore provides reduced complexity at the cost of reduced dynamic capability.

2.3 General Representation

In Part II, we present a method that constructs a system input that would steer the vehicle along a path with low conflict probability. This method accepts state space or maneuver automaton models. In order to simplify the notation, we now present a unified representation which incorporates both the state space representation of Section 2.1 and the maneuver automaton representation of Section 2.2.

Let the input to the system be defined as

$$\boldsymbol{\mu} : [t_0, t_f] \to \mathcal{V}, \tag{2.13}$$

where $\mathcal{V}$ is the input space, which is given by $\mathcal{U}$ for the state space model and by the maneuver space $\mathcal{M}$[1]. Let the nominal vehicle states $\mathbf{y}(t)$ for the input $\boldsymbol{\mu}(t)$ and initial nominal states $\mathbf{y}(t_0)$ be given by the solution

$$\mathbf{y}(t) = \varphi_{\boldsymbol{\mu}}\left(\mathbf{y}(t_0), t\right), \tag{2.14}$$

where the continuous vehicle states $\mathbf{x}(t) \subseteq \mathbf{y}(t)$. Similar to Equation 2.2, the differential constraints are imposed by an inequality:

$$G(\mathbf{y}(t), \boldsymbol{\mu}(t)) \leq \mathbf{0}. \tag{2.15}$$

[1] See Subsection 6.2.1 for a definition of the maneuver space.

When uncertainty in the system is taken into account, the vehicle states are modelled as a stochastic process $\mathbf{Y}(t,\omega)$ and the disturbance is given by $\mathbf{W}(t,\omega)$, Equation 2.14 becomes

$$\mathbf{Y}(t,\omega) = \varphi_{\boldsymbol{\mu},\mathbf{W}(\omega)}\left(\mathbf{Y}(t_0,\omega),t\right). \tag{2.16}$$

We use this general representation when referring to a model that could be formulated either according to the state space representation in Section 2.1 or the maneuver automaton representation in Section 2.2.

2.4 Modelling Autonomous Underwater Vehicles

The funding for the research presented in this thesis have been provided with the AUV application in mind. However, the methods are applicable to various types of autonomous vehicles. As an example, we now present a brief overview of autonomous underwater vehicles and then show that they can be modelled according to the preceding sections.

Most unmanned underwater vehicles in use are remotely operated vehicles (ROVs). These vehicles are tethered and continuously controlled by operators. In the past two decades, there has been much research interest in and development of unmanned underwater vehicles without tethers or operators, called autonomous underwater vehicles (AUVs). ROVs have some advantages over AUVs, including easy and reliable power supply and communications via the tether and the use of sophisticated manipulators. AUVs generally offer more advantages, which include low operational costs, bigger range and no need for a surface support craft or an operator.

The current applications for AUVs include the survey and mapping of the ocean floor for scientific and mining applications, inspection of undersea structures, environmental monitoring, mine countermeasures and research testbeds. Potential future applications include geological sampling, off-board sensors for submarines, construction and maintenance of undersea structures, disposal of mines, inspection of nuclear power plants and commercial salvaging [13, 87].

The sensors used in the navigation and control of AUVs comprise various types of sonars, optical sensors (including camera and laser sensors), inertial measurement units (IMUs, including rate gyros, accelerometers and magnetometers), GPS and pressure sensors [13, 87]. The GPS measurements can only be made when the AUV is at the surface or when the relative position to a surface beacon with a GPS sensor can be determined with acoustic positioning.

The information received from the sensors is used to estimate the vehicle and environmental states. This is an active research area and is called simultaneous localisation and mapping (SLAM) with detection and tracking of moving obstacles (DATMO) [28, 79]. Two of the most common filters used in the estimation are the Extended Kalman Filter (EKF) and the Unscented Kalman Filter (UKF). In both these filters the estimated states are either assumed to be normally distributed or approximated with a normal distribution. When such an estimator is used, the predicted vehicle and environment states are also normally distributed, conforming to the general model in Subsection 2.1.

The actuators on an AUV usually consist of thrusters and control surfaces or fins. An AUV has six degrees of freedom: three angular states (roll, pitch and yaw) and three translational states (surge, sway and heave). The control surfaces usually provide three degrees of control whereas an AUV often has only one thruster which provide one degree of control – AUVs are therefore often underactuated.

The open-loop model of an AUV is generally highly non-linear and coupled [20] where the non-linearities are mostly caused by a coordinate transformation between the body-fixed and inertial reference frames, non-linear actuator dynamics and hydrodynamics. The conventional way to design a controller is to linearise the AUV model and use linear control techniques such as a set of SISO PID-controllers, linear quadratic optimal control [52], robust control, successive loop closure [20] or time-varying linear control such as gain-scheduling [74]. The closed-loop

models of the systems using these controllers can be modelled as time-variant linear systems and conform to the description of Equation 2.7. Together with the assumption of Gaussian distributed disturbance inputs employed by the commonly-used Kalman filter, the prediction of the mean and covariance of vehicle states is straightforward, as stated by Subsection 2.1.

More advanced non-linear controllers are also used for AUV control. Of these, state feedback linearisation [20] ensures that the closed-loop system is linear which then conforms to the system of Equation 2.7. Other non-linear controllers such as sliding-mode control [20], switched seesaw control [1], backstepping control [2, 16, 69] and feedback passivation [40] do not conform to the system of Equation 2.7. However, it might still be possible to approximate the closed-loop system sufficiently accurately by Equations 2.4 and 2.5, in which case the methods presented in this thesis are applicable.

Although some controller designs might cause the conflict detection and resolution methods presented in this thesis not to be applicable to the closed-loop system, there are several good applicable controller designs available. The overall system design will then dictate the set of controller designs to choose from.

We now have a general representation for the vehicle dynamics and have motivated that autonomous underwater vehicles conform to this representation. We use this vehicle model for the development of methods for conflict detection (Part I) and conflict resolution (Part II).

Part I

Conflict Detection

Chapter 3

Overview of Conflict Detection

Part I present the development of a novel probabilistic conflict detection method. This chapter introduces important concepts in conflict detection and gives an overview of existing conflict detection methods. Section 3.1 details the different approaches to conflict detection and motivates the choice of the probabilistic approach which is used in this thesis. Section 3.2 formally defines the general probabilistic conflict detection problem and Section 3.3 gives an overview of existing probabilistic conflict detection methods.

3.1 Approaches to Conflict Detection

According to the survey of conflict detection and resolution methods by Kuchar and Yang [43, 44], the different approaches to conflict detection can be discriminated by the way the vehicle and environment states are projected into the future. We now give an overview of these approaches and motivate our choice of the probabilistic approach.

The *nominal projection method* only propagates the most likely vehicle and environment states into the future along a single relative trajectory. This approach therefore ignores any uncertainty in the state projections. It is usually simple to propagate the states and to check for conflict, but the conflict detection method could miss likely conflict conditions, especially with much uncertainty in the system and when the propagation period is extended.

The *worst-case projection method* propagates the possible range of the vehicle and environment states into the future. These ranges of states are then viewed as regions in space that should not overlap. It is usually fairly simple to construct these regions and check for conflict, but this approach tends to predict that conflict will occur even when the probability of conflict is very low. It is therefore an overly cautious approach which does not perform well in systems with much uncertainty or when the propagation period is extended.

The *probabilistic projection method* strikes a balance between the nominal and worst-case approaches. It assigns probability distributions to the vehicle and environment states and propagates these probability distributions into the future. In this case conflict detection does not entail giving a binary prediction of conflict or no conflict, but rather calculating the probability of conflict. The vehicle trajectories with a probability of conflict below a certain threshold are then considered safe. As the probabilistic approach suffers less from missed detection than the nominal approach and suffers less from false alarms than the worst-case approach, it is the desired approach taken in this thesis. Methods using the probabilistic approach are usually computationally expensive and the focus of research into these methods is therefore on improving their efficiency in order to calculate the probability of conflict in real-time.

3.2 Probabilistic Problem Definition

In this section, we formally state the probabilistic conflict detection problem and define the environment models applicable to the conflict detection method proposed in this thesis. Subsection 3.2.1 presents the basic problem definition for a point mass with uncertain states and a known conflict region. This is quite a restrictive definition, therefore Subsection 3.2.2 focuses on methods of transforming a more general and useful problem description to the basic problem definition. Lastly, Subsection 3.2.3 explains how multiple obstacle regions with different uncertainty parameters can be managed using the basic problem definition.

3.2.1 Basic Problem Definition

For the basic conflict detection problem definition, we have a vehicle modelled as a point mass with uncertain states that should not enter a known conflict region. Let the position of the vehicle be described by the stochastic process $\mathbf{R}(t, \omega)$ where $t \in [t_0, t_f]$ is the time variable and $\omega \in \Omega$ is the outcome with Ω the sample space or set of all possible outcomes [64]. The outcome ω can be thought of as an index to a possible vehicle trajectory. Notably, $\mathbf{R}(t, \omega)$ describes the vehicle position in the absence of conflict and is therefore invariant to the conflict region description. Let C_t be the conflict region indexed at time t and defined as the set of positions that the vehicle may not occupy. A conflict condition therefore exists whenever $\mathbf{R}(t, \omega) \in C_t$. C_t is allowed to change over time, but assumed to contain no uncertainty. We want to calculate the probability of conflict for a future time period $[t_0, t_f]$. In order to do this, we define the set of outcomes for which a conflict condition exists during $[t_0, t]$ as

$$A_t \triangleq \{\omega \in \Omega : \exists \tau \in [t_0, t], \mathbf{R}(\tau, \omega) \in C_\tau\}. \tag{3.1}$$

The set of all outcomes for which no conflict condition occurs during $[t_0, t]$, which is the complement of A_t, is given by

$$B_t \triangleq \{\omega \in \Omega : \forall \tau \in [t_0, t], \mathbf{R}(\tau, \omega) \notin C_\tau\}. \tag{3.2}$$

The probability of conflict during $[t_0, t_f]$ is now defined as

$$P_C(t_f) \triangleq P\left[A_{t_f}\right] = 1 - P\left[B_{t_f}\right]. \tag{3.3}$$

The probability of conflict is therefore the probability that a vehicle trajectory will enter the conflict region at least once during the interval in question. This definition is consistent with the work of Kuchar and Yang [81–84], Paielli and Erzberger [55, 56] and Jones [37, 38].

Another conflict probability definition used by some researchers is the maximum instantaneous conflict probability during a specified time period, or

$$\begin{aligned} P_C^*(t_f) &\triangleq \max_{\tau \in [t_0, t_f]} P\left[\{\omega \in \Omega : \mathbf{R}(\tau, \omega) \in C_\tau\}\right] \\ &\leq P_C(t_f). \end{aligned} \tag{3.4}$$

This definition was used by Prandini et al. [66–68] and Fulgenzi et al. [27]. It can be shown to be a lower bound to the probability of conflict defined in Equation 3.3. The maximum instantaneous probability of conflict $P_C^*(t)$ of Equation 3.4 is much easier to compute than the probability of conflict $P_C(t)$ as defined in Equation 3.3, but using $P_C^*(t)$ is unsafe because it underestimates the actual conflict probability, especially when the vehicle spends a prolonged time near the conflict region. We therefore only use the definition in Equation 3.3.

3.2.2 Transformation from General Formulation

The definition of conflict probability of Subsection 3.2.1 is only applicable to a vehicle that can be approximated as a point mass and a conflict region without uncertainty. In this subsection, we show how a more general and useful problem formulation can be transformed to fit that of Subsection 3.2.1.

For the general formulation, we assume that the vehicle and obstacles are contained in regions[1] that should not intersect. All these regions are allowed to have uncertain positions and velocities and their shape and orientation are assumed to be variable but known. A safety zone can be added to either region – the vehicle or obstacle conflict region can be any shape as long as it contains the region occupied by the actual vehicle or obstacle. Furthermore, the position of the obstacle conflict region does not need to be fixed and the obstacle conflict region can consist of multiple sections. This means that the obstacle conflict region can model multiple moving vehicles.

Let the vehicle conflict region at time t be given by C_t^v and $p_v \in C_t^v$ be an arbitrary point. Choose any point $c_v \in C_t^v$ to be a reference point. Let the position of c_v be given by the stochastic process $\mathbf{R}_c^v(t,\omega)$ and the position of p_v relative to c_v be known and given by the vector $\mathbf{r}_{p/c}^v(t)$. An illustration of this setup in three dimensions is shown in Figure 3.1. Similarly,

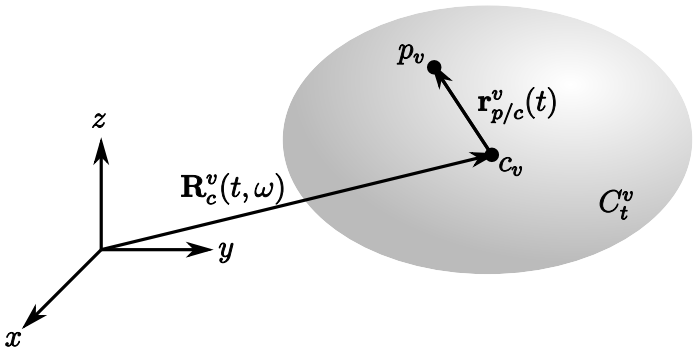


Figure 3.1: Vehicle conflict volume

let the obstacle conflict region at time t be given by C_t^o and $p_o \in C_t^o$ be an arbitrary point. Choose $c_o \in C_t^o$ to be the reference point. Let the position of c_o be given by the stochastic process $\mathbf{R}_c^o(t,\omega)$ and the position of p_o relative to c_o be given by the known vector $\mathbf{r}_{p/c}^o(t)$. The set of outcomes for which a conflict condition exists during $[t_0,t]$ is then given by

$$A_t = \{\omega \in \Omega : \exists \tau \in [t_0,t], Q(\tau,\omega)\} \tag{3.5}$$

where the predicate $Q(\tau,\omega)$ is given by

$$\begin{aligned} Q(\tau,\omega) &= \exists p_v \in C_\tau^v,\, \exists p_o \in C_\tau^o,\, \mathbf{R}_c^v(\tau,\omega) + \mathbf{r}_{p/c}^v(\tau) = \mathbf{R}_c^o(\tau,\omega) + \mathbf{r}_{p/c}^o(\tau) \\ &= \mathbf{R}_c^v(\tau,\omega) - \mathbf{R}_c^o(\tau,\omega) \in \left\{ \mathbf{r}_{p/c}^o(\tau) - \mathbf{r}_{p/c}^v(\tau) : p_v \in C_\tau^v \wedge p_o \in C_\tau^o \right\}. \end{aligned} \tag{3.6}$$

Following this, if we set

$$\mathbf{R}(\tau,\omega) = \mathbf{R}_c^v(\tau,\omega) - \mathbf{R}_c^o(\tau,\omega) \tag{3.7}$$

and

$$C_\tau = \left\{ \mathbf{r}_{j/c}^o(\tau) - \mathbf{r}_{i/c}^v(\tau) : p_v \in C_\tau^v \wedge p_o \in C_\tau^o \right\} \tag{3.8}$$

[1] A region refers to a volume in three dimensions and an area in two dimensions.

in Equation 3.6, Equation 3.5 reduces to Equation 3.1, which proves that the more general problem formulation of this subsection can be transformed to that of Subsection 3.2.1.

In general, the probability density function of $\mathbf{R}(\tau, \omega)$ in Equation 3.7 is calculated by the convolution of the probability density functions of $\mathbf{R}_c^v(\tau, \omega)$ and $-\mathbf{R}_c^o(\tau, \omega)$ [64] which is a computationally expensive calculation. However, when the probability density functions of $\mathbf{R}_c^v(\tau, \omega)$ and $\mathbf{R}_c^o(\tau, \omega)$ are Gaussian and independent, the probability density function of $\mathbf{R}(\tau, \omega)$ is also Gaussian of which the mean and covariance are simply calculated by summation [64]. As motivated in Section 2.4, the positions and velocities of vehicles and obstacles are often modelled as having Gaussian distributions, causing the calculation of Equation 3.7 to be inexpensive.

There is no general method of constructing the conflict region C_τ according to Equation 3.8 – therefore this calculation has to be handled case by case. However, the construction of the conflict region is simplified significantly if vehicle region shapes are chosen that are invariant under rotation such as circles (for the two-dimensional case) or spheres (for the three-dimensional case) and if we assume that the shapes of the vehicle and obstacle conflict regions stay constant over the prediction period. A conceptual illustration of the process of constructing the conflict region from a vehicle and obstacle conflict region is shown in Figure 3.2. Some

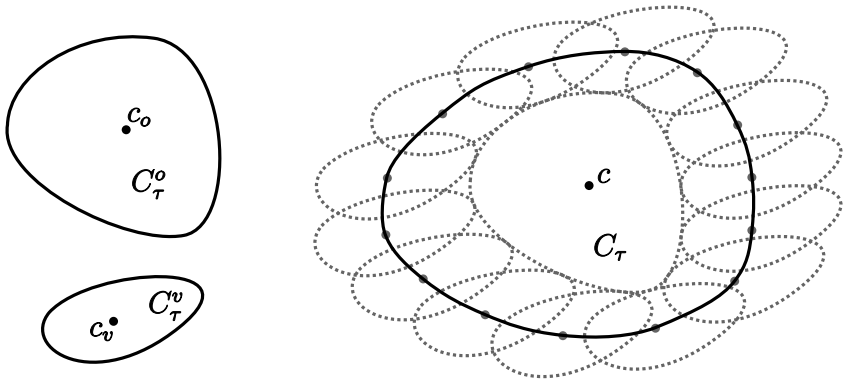


Figure 3.2: Construction of C_τ form C_τ^v and C_τ^o

examples of conflict regions and their construction are given in Chapter 5.

3.2.3 Handling Multiple Obstacle Regions

When the obstacle region consists of multiple distinct sections, the problem formulation in Subsection 3.2.2 can only be used when the shapes of all the probability density functions associated with the distinct sections are the same. This proves to be a limiting restriction when the environment consists of multiple moving obstacles. In order to manage these types of environments, we now show how the combined conflict probability can be calculated by the sum of the conflict probabilities due to individual sections of the obstacle region.

We illustrate here how to manage two distinct sections of the obstacle region. This process is similar for more than two sections. Let the one section of the obstacle region at time t be denoted by C_t^{o1}, the position of a reference point $c_{o1} \in C_t^{o1}$ be given by $\mathbf{R}_c^{o1}(t, \omega)$ and the position of an arbitrary point $p_{o1} \in C_t^{o1}$ relative to c_{o1} be given by $\mathbf{r}_{p/c}^{o1}(t)$. Similarly, let the other section of the obstacle region at time t be denoted by C_t^{o2}, the position of a reference point $c_{o2} \in C_t^{o2}$ be given by $\mathbf{R}_c^{o2}(t, \omega)$ and the position of an arbitrary point $p_{o2} \in C_t^{o2}$ relative to c_{o2} be given by $\mathbf{r}_{p/c}^{o2}(t)$. The vehicle region is defined as in Subsection 3.2.2. If we define the predicate that specifies a conflict condition due to the obstacle region section C_t^{o1} as

$$Q_1(\tau, \omega) = \mathbf{R}_c^v(\tau, \omega) - \mathbf{R}_c^{o1}(\tau, \omega) \in \left\{ \mathbf{r}_{p/c}^{o1}(\tau) - \mathbf{r}_{p/c}^v(\tau) : p_v \in C_\tau^v \wedge p_{o1} \in C_\tau^{o1} \right\} \tag{3.9}$$

and the predicate that defines a conflict condition due to the obstacle region section C_t^{o2} as

$$Q_2(\tau,\omega) = \mathbf{R}_c^v(\tau,\omega) - \mathbf{R}_c^{o2}(\tau,\omega) \in \left\{ \mathbf{r}_{p/c}^{o2}(\tau) - \mathbf{r}_{p/c}^{v}(\tau) : p_v \in C_\tau^v \wedge p_{o2} \in C_\tau^{o2} \right\}, \tag{3.10}$$

the set of all outcomes for which a conflict condition occurs in the time period $[t_0, t]$ is given by

$$\begin{aligned} A_t &= \{\omega \in \Omega : (\exists \tau_1 \in [t_0,t], Q_1(\tau_1,\omega)) \vee (\exists \tau_2 \in [t_0,t], Q_2(\tau_2,\omega))\} \\ &= \{\omega \in \Omega : \exists \tau \in [t_0,t], Q_1(\tau,\omega)\} \cup \{\omega \in \Omega : \exists \tau \in [t_0,t], Q_2(\tau,\omega)\} \\ &= A_{t1} \cup A_{t2} \end{aligned} \tag{3.11}$$

where A_{t1} and A_{t2} are the sets of outcomes for which a conflict condition occurs due to the obstacle region sections C_t^{o1} and C_t^{o2} respectively. The probability of conflict is then calculated according to Equation 3.3 as

$$\begin{aligned} P\left[A_{t1} \cup A_{t2}\right] &= P\left[A_{t1}\right] + P\left[A_{t2}\right] - P\left[A_{t1} \cap A_{t2}\right] \\ &\leq P\left[A_{t1}\right] + P\left[A_{t2}\right]. \end{aligned} \tag{3.12}$$

We can therefore calculate an upper bound to the probability of conflict by summing the probability of conflict that resulted from the individual sections of the obstacle region. A low threshold value that defines the highest probability of conflict that is considered safe is usually selected. For low values of conflict probability, the probability that a vehicle trajectory will enter *both* the obstacle regions, $P\left[A_{t1} \cap A_{t2}\right]$, can reasonably be expected to be very low. The upper bound calculated in Equation 3.12 can therefore be expected to be tight for conflict probabilities at or below the threshold value. For higher conflict probability values, the upper bound might not be a tight one, but since the actual conflict probability is higher than the threshold, it would not affect the outcome.

3.3 Overview of Existing Methods

Having defined the probabilistic conflict detection problem, we give an overview of existing probabilistic conflict detection methods.

A well-known earlier method is due to Paielli and Erzberger [55, 56]. They initially derived an analytic solution for two airplanes with crossing flight paths in two dimensions where the conflict zone is given by a circle, the aircraft positions are normally distributed and the aircraft velocities are assumed constant [55]. They later extended this method by deriving an approximate analytic solution for three dimensions [56]. Hwang et al. [34] extended the two-dimensional method to include, together with the constant velocity mode, a coordinated turn mode for each aircraft.

Patera [57–62] devised a number of methods to calculate the probability of collision for satellites. For all these methods, the relative position between the satellite and obstacle is assumed to be normally distributed, but the relative velocity is assumed to be known. Some methods assume constant relative velocity during the encounter [57, 59], but allow, in contrast to the methods of Paielli and Erzberger [55, 56], arbitrary conflict zone shapes. The probability of collision is then calculated by numerical integration of a contour integral. These methods were extended for nonlinear relative velocity for arbitrary conflict volume shapes [58] and simplified for spherical [61] and ellipsoidal [62] conflict volumes. The methods for nonlinear relative velocity assume that the conflict volume is small relative to the distribution of the relative position. Patera proposed a method [60] for cases where this assumption would cause significant error. He also introduced a concept similar to probability flow, which is introduced in Chapter 4, to decrease the computational complexity [58].

The methods of Yang and Kuchar [81, 82, 84] are based on Monte Carlo simulations. Although they only applied the Monte Carlo simulation technique to the specific problem of

computing the probability of conflict for commercial airspace, their method can easily be extended to deal with a general problem with non-Gaussian probability distributions, complex vehicle maneuvers and arbitrary conflict zone shapes. However, Monte Carlo simulations are computationally expensive, therefore Yang and Kuchar resorted to creating lookup tables for a discrete number of scenarios based on off-line Monte Carlo simulations [82] or approximating the aircraft trajectories by a small number of straight-line segments [81, 84] in order to compute the probability of conflict in real-time with a satisfactory degree of accuracy.

A method created by Jones [37, 38] allows Gaussian uncertainty in all the vehicle and obstacle states for ellipsoidal conflict regions. The probability of conflict is calculated by applying a one-dimensional metric and then integrating numerically. The method allows any vehicle trajectory, as long as the mean and covariance of the vehicle states are available at the required time instances. He also introduced the idea of calculating an upper bound to the probability of conflict in order to reduce the computational complexity.

The preceding overview shows that there exists a trade-off between the computational complexity and the generality of the models in probabilistic methods: more simplifying assumptions make the probability of conflict easier to compute. There are no existing methods that can compute the conflict probability in real-time for complex vehicle trajectories, cluttered environments and a variety of conflict region shapes. The next chapter presents a novel probabilistic conflict detection method for arbitrary vehicle and obstacle trajectories and conflict region shapes, using only the assumption of Gaussian distributed states. We also show that the probability of conflict can be calculated in real-time, even for cluttered environments by introducing the concept of probability flow. This method can be viewed as a generalisation of the above-mentioned methods, excluding the Monte Carlo simulation methods.

Chapter 4

Conflict Detection using Probability Flow

With the definition of the probability of conflict of Chapter 3 established previously, the focus of this thesis now shifts to finding an efficient method to calculate the probability of conflict. A possible manner in which to calculate this is to propagate the probability density function of the vehicle position into the future and accumulate the overlap between the probability density function and the conflict region at incremental time instants, which is the approach taken by Jones [37, 38]. The conflict region distorts the probability density function, impacting on the computational load in two ways: firstly, the propagation of the distorted probability function is computationally complex and secondly, the overlap between the distorted probability function and the conflict region is not easily calculated.

The basic concept of the most important contribution of this thesis is viewing the conflict detection problem as the accumulation of the rate of probability increase at the *boundary* of the conflict region, instead of the accumulation of the overlap between the probability density function and the *interior* of the conflict region. The advantage of this approach is twofold: firstly, the accumulation is done across the boundary instead of across the interior of the conflict region, reducing the dimensions of the problem by one, and secondly, for reasons detailed later in this chapter, the probability density function of the vehicle states can be assumed to be unaffected by the conflict region, making propagation of the probability density function efficient.

We term the rate of conflict probability increase *probability flow*. A similar concept is found in a paper by Patera [58] who also used it to increase the efficiency of the conflict probability calculation. The concept of flow of probability space into a hazard was also mentioned by Jones [37]. However, this derivation of an expression for probability flow and the generality of the problem definition combine to form a novel and useful conflict detection method that improves significantly on existing methods. The rest of this chapter is adapted from a recently published paper by the author [76].

This chapter consists of two parts: the mathematical derivation of an expression for the probability flow and a discussion of the simplifying assumptions and numerical techniques necessary for real-time conflict probability calculation. Section 4.1 formally defines probability flow which is used in Section 4.2 to derive an expression for the probability flow at the conflict region boundary. The two simplifying assumptions, namely a close upper bound calculation and Gaussian distributed vehicle states, are discussed in Sections 4.3 and 4.4 respectively, and the adaptive integration procedure used to integrate numerically is detailed in Section 4.5.

4.1 Definition of Probability Flow

We define *probability flow* as the instantaneous rate of increase in the probability of conflict, or $\frac{\mathrm{d}P_C(t)}{\mathrm{d}t}$. This can be interpreted as the rate of increase in the probability that the projected vehicle trajectories enter the conflict volume, hence the term *probability flow*. The probability of conflict for the interval $[t_0, t_f]$ can then be calculated by using[1]

$$P_C\left(t_f\right) = \int_{t_0}^{t_f} \frac{\mathrm{d}P_C(\tau)}{\mathrm{d}\tau}\,\mathrm{d}\tau. \tag{4.1}$$

We now have to find an expression for the probability flow. The first step in this quest is to find an expression for the change in A_t in Equation 3.1 with respect to time. The set of outcomes for which a conflict condition exists during $[t_0, t + \Delta t]$ is given by

$$\begin{aligned} A_{t+\Delta t} = &\ \{\omega \in \Omega : (\exists \tau \in [t_0, t], \mathbf{R}(\tau, \omega) \in C_\tau) \\ &\vee (\exists \nu \in [t, t + \Delta t], \mathbf{R}(\nu, \omega) \in C_\nu)\} \\ = &\ \Delta A_{t:t+\Delta t} \cup A_t, \end{aligned} \tag{4.2}$$

where

$$\Delta A_{t:t+\Delta t} \triangleq \{\omega \in \Omega : \exists \tau \in [t, t + \Delta t], \mathbf{R}(\tau, \omega) \in C_\tau\}. \tag{4.3}$$

The increase in conflict probability during $[t, t + \Delta t]$ is now given by

$$\begin{aligned} P\left[A_{t+\Delta t}\right] - P\left[A_t\right] &= P\left[(\Delta A_{t:t+\Delta t} \cup A_t) \cap (A_t \cup B_t)\right] - P\left[A_t\right] \\ &= P\left[(\Delta A_{t:t+\Delta t} \cap B_t) \cup A_t\right] - P\left[A_t\right] \\ &= P\left[\Delta A_{t:t+\Delta t} \cap B_t\right], \end{aligned} \tag{4.4}$$

where $(\Delta A_{t:t+\Delta t} \cap B_t) \cap A_t = \emptyset$. The *probability flow* from B_t to A_t is therefore given by

$$\frac{\mathrm{d}P_C(t)}{\mathrm{d}t} = \lim_{\Delta t \to 0} \left(\frac{1}{\Delta t} P\left[\Delta A_{t:t+\Delta t} \cap B_t\right] \right). \tag{4.5}$$

Equation 4.5 gives an expression for the probability flow in terms of sets of outcomes and is not a very useful formulation. The next section shows how the probability flow can be written in terms of the boundary of the conflict region and the probability density function of the vehicle states.

4.2 Probability Flow through the Conflict Volume Surface

This section derives an expression for the probability flow across the boundary of the conflict region. The derivation is done for the three-dimensional case where the conflict region is a volume and the conflict region boundary is a surface. The derivation for the two-dimensional case is similar and the result is given at the end of this section. We start the derivation by finding the probability flow through a small rectangular element on the conflict volume surface after which we partition the conflict volume surface and sum the probability flow through all the elements in the partition to obtain the total probability flow.

Consider a small flat rectangular element, ΔS, located on the conflict volume surface. We now define the conditions under which the vehicle trajectories that are located outside the conflict volume and are in the vicinity of ΔS, will enter the conflict volume during $[t, t + \Delta t]$. Consider a rectangular column ΔV on ΔS[2] extending away from the conflict volume, shown in Figure 4.1. Assume for the time being that the rest of the boundary of the conflict volume C_t

[1] We require $P_C(\tau)$ to be absolutely continuous on $[t_0, t_f]$. We also assume that the instantaneous conflict probability at t_0 is negligible.

[2] Assume for the time being that the conflict volume surface does not change over time. This assumption is relaxed later in this section.

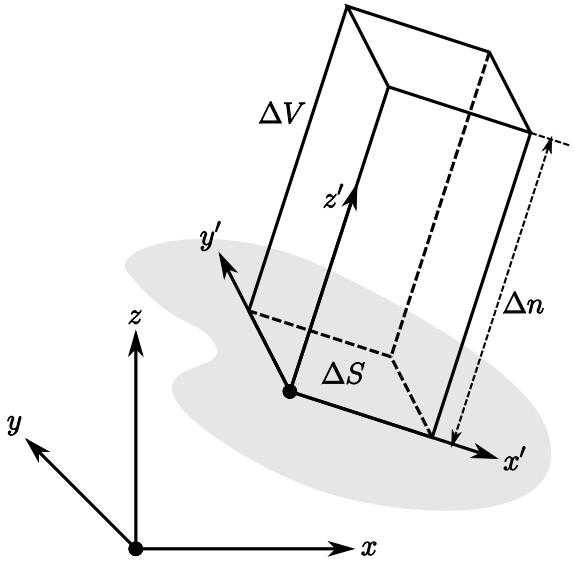


Figure 4.1: Diagram depicting the column ΔV on the rectangular element ΔS

is given by the plane containing ΔS[3]. The rate of increase in $P_C(t)$ due to vehicle positions in ΔV at time t is given by

$$\frac{\mathrm{d}P_C^{\Delta S}(t)}{\mathrm{d}t} = \lim_{\Delta t \to 0} \left(\frac{1}{\Delta t} P\left[\Delta A_{t:t+\Delta t} \cap B_t \cap D_t\right] \right), \tag{4.6}$$

where $D_t \triangleq \{\omega \in \Omega : \mathbf{R}(t,\omega) \in \Delta V\}$. Suppose the vehicle velocity state is given by the stochastic process $\mathbf{V}(t,\omega)$ and the unit vector normal to ΔS and pointing away from the conflict volume C_t is given by $\mathbf{n}$. We define the velocity component normal to the surface element ΔS as

$$V_n(t,\omega) \triangleq \mathbf{n} \cdot \mathbf{V}(t,\omega). \tag{4.7}$$

We also define a closely related stochastic process as

$$V_n'(t : t+\Delta t, \omega) \triangleq \frac{1}{\Delta t} \min_{\zeta \in [0,\Delta t]} \left[\int_t^{t+\zeta} V_n(\tau,\omega)\,\mathrm{d}\tau \right]. \tag{4.8}$$

When $V_n(t,\omega) < 0$, $V_n'(t : t+\Delta t, \omega) \to V_n(t,\omega)$ as $\Delta t \to 0$.

Now define a partition $P_v(a) = \{v_0, v_1, \ldots, v_n\}$ on the interval $[0, a]$ with $a > 0$. For any $v_i \in P_v(a)$, define ΔV_i as the column on ΔS with height $\Delta n_i = v_i \Delta t$ and $\Delta E_{i-1:i} \triangleq \Delta V_{i-1} \cap \Delta V_i$. These definitions are shown in Figure 4.2. If $-V_n'(t : t+\Delta t, \omega) \in [v_{i-1}, v_i]$, then all $\mathbf{R}(t,\omega) \in \Delta V_{i-1}$ and some $\mathbf{R}(t,\omega) \in \Delta E_{i-1:i}$ enter the conflict volume during $[t, t+\Delta t]$. If we define

$$G_{t:t+\Delta t}^{i-1:i} \triangleq \left\{\omega \in \Omega : v_{i-1} \le -V_n'(t : t+\Delta t, \omega) \le v_i\right\} \tag{4.9}$$

and

$$D_t^i \triangleq \{\omega \in \Omega : \mathbf{R}(t,\omega) \in \Delta V_i\}, \tag{4.10}$$

then we can write

$$\begin{aligned} P\left[\Delta A_{t:t+\Delta t} \cap B_t \cap D_t\right] = \lim_{a \to \infty} \sum_{i=1}^{n} \big(& P\left[G_{t:t+\Delta t}^{i-1:i} \cap D_t^{i-1} \cap B_t\right] \\ & + \alpha_i P\left[G_{t:t+\Delta t}^{i-1:i} \cap D_t^{i-1} \cap D_t^i \cap B_t\right] \big), \end{aligned} \tag{4.11}$$

[3]This assumption is also relaxed later in this section.

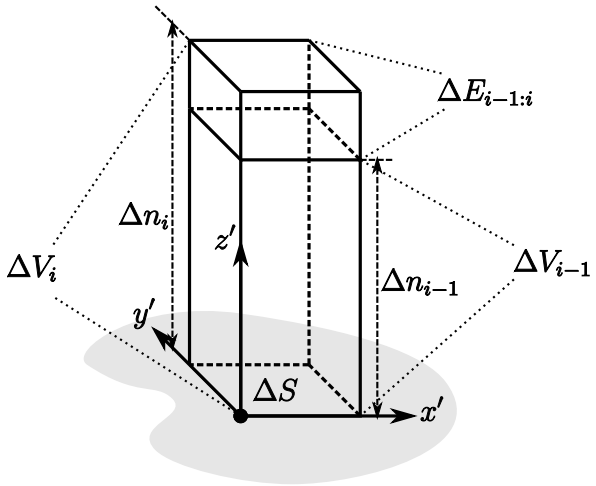


Figure 4.2: Diagram depicting ΔV_{i-1}, ΔV_i and $\Delta E_{i-1:i}$

where $\alpha_i \in [0,1]$. When $\|P_v(a)\| \to 0$, the volume of $\Delta E_{i-1:i}$ tends to zero and the last term of the sum in Equation 4.11 disappears. Using Bayes' Theorem and the properties of probability density functions, Equation 4.11 becomes

$$\begin{aligned} P\left[\Delta A_{t:t+\Delta t} \cap B_t \cap D_t\right] &= \lim_{a\to\infty} \lim_{\|P_v(a)\|\to 0} \sum_{i=1}^{n} \Big(P\left[G_{t:t+\Delta t}^{i-1:i} | D_t^{i-1} \cap B_t\right] \times \\ &\quad P\left[D_t^{i-1} | B_t\right] \Big) P\left[B_t\right] \\ &= \lim_{\|P_v\|\to 0} \lim_{a\to\infty} \sum_{i=1}^{n} \Bigg(\int_{-v_i}^{-v_{i-1}} f_{V_n'(t,\Delta t)}^{B_t}\left(v_n | D_t^i\right) \, \mathrm{d}v_n \times \\ &\quad \iiint_{V_{i-1}} f_{\mathbf{R}(t)}^{B_t}(\mathbf{p}) \, \mathrm{d}V \Bigg) \left(1 - P_C(t)\right) \\ &= \left(1 - P_C(t)\right) \int_{-\infty}^{0} \Bigg(f_{V_n'(t,\Delta t)}^{B_t}\left(v_n | D_t^{v_n}\right) \times \\ &\quad \iiint_{V(v_n)} f_{\mathbf{R}(t)}^{B_t}(\mathbf{p}) \, \mathrm{d}V \Bigg) \mathrm{d}v_n, \end{aligned} \tag{4.12}$$

where $f_{\mathbf{R}(t)}(\cdot)$ and $f_{V_n'(t,\Delta t)}(\cdot)$ are the probability density functions of $\mathbf{R}(t,\omega)$ and $V_n'(t : t+\Delta t, \omega)$ respectively, $f_{V_n'(t,\Delta t)}^{B_t}(v_n) \triangleq f_{V_n'(t,\Delta t)}(v_n | B_t)$, $\Delta V(v_n)$ is the column on ΔS with height $-v_n \Delta t$, $D_t^{v_n} \triangleq \{\omega \in \Omega : \mathbf{R}(t) \in \Delta V(v_n)\}$ and $\mathbf{p}$ is the integration variable over $\mathrm{d}V$. Suppose that the vehicle states are defined in the (x, y, z)-coordinate system, the (x', y', z')-coordinate system is anchored to ΔV as shown in Figure 4.2 and the transformation from the latter to the former coordinates is given by $T(\cdot)$. When the volume integral of Equation 4.12 is written as an iterated

integral, we write

$$\begin{aligned}\frac{1}{\Delta t}\iiint\limits_{V(v_n)} f^{B_t}_{\mathbf{R}(t)}(\mathbf{p})\,\mathrm{d}V &= \frac{1}{\Delta t}\iint\limits_{\Delta S}\int_0^{-v_n\Delta t} f^{B_t}_{\mathbf{R}(t)}\left(T\left(x',y',z'\right)\right)\,\mathrm{d}z'\,\mathrm{d}S \\ &= -v_n\iint\limits_{\Delta S} f^{B_t}_{\mathbf{R}(t)}\left(T\left(x',y',0\right)\right)\,\mathrm{d}S \\ &\quad+\frac{1}{\Delta t}\iint\limits_{\Delta S}\int_0^{-v_n\Delta t}\Big[f^{B_t}_{\mathbf{R}(t)}\left(T\left(x',y',z'\right)\right) \\ &\quad- f^{B_t}_{\mathbf{R}(t)}\left(T\left(x',y',0\right)\right)\Big]\,\mathrm{d}z'\mathrm{d}S.\end{aligned} \tag{4.13}$$

If $f^{B_t}_{\mathbf{R}(t)}(\cdot)$ contains no discontinuities, the last term of Equation 4.13 disappears when $\Delta t \to 0$. Equation 4.6 now becomes

$$\begin{aligned}\frac{\mathrm{d}P_C^{\Delta S}(t)}{\mathrm{d}t} = &-(1-P_C(t))\iint\limits_{\Delta S} f^{B_t}_{\mathbf{R}(t)}(\mathbf{p})\,\mathrm{d}S\times \\ &\lim_{\Delta t\to 0}\int_{-\infty}^{0} f^{B_t}_{V_n'(t,\Delta t)}\left(v_n|D_t^{v_n}\right)v_n\,\mathrm{d}v_n.\end{aligned} \tag{4.14}$$

When $\Delta t \to 0$, the height of $\Delta V(v_n)$ tends to zero for any v_n. To see this, divide the integration interval over v_n into $[0,b]$ and $[b,-\infty)$ for some finite $b<0$. For $v_n\in[0,b]$, the height of $\Delta V(v_n)$ clearly tends to zero as $\Delta t\to 0$. If we require the first moment of the conditional distribution of v_n to exist for any height of $\Delta V(\cdot)$, then there exists a finite $\mu_1^-(0)$ such that

$$\begin{aligned}\mu_1^-(0) &\triangleq \int_{-\infty}^{0} f^{B_t}_{V_n(t)}\left(v_n|D_t^{v_k}\right)v_n\,\mathrm{d}v_n \\ &= \lim_{b\to-\infty}\int_b^0 f^{B_t}_{V_n(t)}\left(v_n|D_t^{v_k}\right)v_n\,\mathrm{d}v_n + \lim_{b\to-\infty}\int_{-\infty}^{b} f^{B_t}_{V_n(t)}\left(v_n|D_t^{v_k}\right)v_n\,\mathrm{d}v_n \\ &= \mu_1^-(0) + \lim_{b\to-\infty}\int_{-\infty}^{b} f^{B_t}_{V_n(t)}\left(v_n|D_t^{v_k}\right)v_n\,\mathrm{d}v_n,\end{aligned} \tag{4.15}$$

which proves that

$$\lim_{b\to-\infty}\int_{-\infty}^{b} f^{B_t}_{V_n(t)}\left(v_n|D_t^{v_k}\right)v_n\,\mathrm{d}v_n = 0 \tag{4.16}$$

for any $v_k\in[0,-\infty)$. Equation 4.14 can now be written as

$$\frac{\mathrm{d}P_C^{\Delta S}(t)}{\mathrm{d}t} = -(1-P_C(t))\iint\limits_{\Delta S} f^{B_t}_{\mathbf{R}(t)}(\mathbf{p})\,\mathrm{d}S\int_{-\infty}^{0} f^{B_t}_{V_n(t)}\left(v_n|D_t^{\Delta S}\right)v_n\,\mathrm{d}v_n, \tag{4.17}$$

where $D_t^{\Delta S}\triangleq\{\omega\in\Omega:\mathbf{R}(t)\in\Delta S\}$. Notice that the height of $\Delta V(v_n)$ in Equation 4.14 has shrunk to zero, therefore all vehicle trajectories in $D_t^{v_n}$ that enter the conflict volume during $[t,t+\Delta t]$ do so only through ΔS and no vehicle trajectories outside $D_t^{v_n}$ enter the conflict volume through ΔS. Equation 4.17 therefore calculates the flow of probability through the surface area ΔS. We are therefore able to relax the assumption that the conflict volume surface is given by the plane containing ΔS – the conflict volume surface outside of ΔS is allowed to be any shape.

With an expression for the probability flow through a rectangular surface area established, we now partition the conflict volume surface into small rectangular areas and, by applying

Equation 4.17, find an expression for the probability flow through the entire conflict volume surface.

Assume the boundary of the region of conflict is given by a piece-wise smooth surface $S(C_t)$ with $\mathbf{n}(\mathbf{p})$ defined as the unit vector normal to $S(C_t)$ at $\mathbf{p}$ and pointing away from C_t. Assume also that the surface remains unchanged over time. This assumption is relaxed at the end of this section. We create a partition $P_S = \{\Delta S_1, \ldots, \Delta S_n\}$ on $S(C_t)$. The probability flow through the entire conflict volume is now given by summing Equation 4.17 over the partition P_S and letting $\|P_S\| \to 0$, yielding

$$\begin{aligned} \frac{\mathrm{d}P_C(t)}{\mathrm{d}t} &= \lim_{\|P_S\|\to 0} \sum_{k=1}^{n} \frac{\mathrm{d}P_C^{\Delta S_k}(t)}{\mathrm{d}t} \\ &= -(1 - P_C(t)) \iint\limits_{S(C_t)} f^{B_t}_{\mathbf{R}(t)}(\mathbf{p}) \int_{-\infty}^{0} f^{B_t}_{V_n(t)}\left(v_n | D_t^{\mathbf{p}}\right) v_n \, \mathrm{d}v_n \, \mathrm{d}S, \end{aligned} \tag{4.18}$$

where $D_t^{\mathbf{p}} \triangleq \{\omega \in \Omega : \mathbf{R}(t, \omega) = \mathbf{p}\}$.

Equation 4.18 is only valid for an unchanging C_t. If the shape of C_t is allowed to change in a continuous manner over time, each surface element ΔS_k of the partition P_S is now allowed to translate, rotate and resize. Let the velocity of a point $\mathbf{p} \in \Delta S_k$ be given by $\mathbf{w}(\mathbf{p}, t)$ and the velocity component normal to the surface of the conflict volume be defined as $w_n(\mathbf{p}, t) \triangleq \mathbf{n}(\mathbf{p}) \cdot \mathbf{w}(\mathbf{p}, t)$. When the derivation of Equation 4.18 is repeated with this relaxed condition, the probability flow through the conflict volume is given by

$$\frac{\mathrm{d}P_C(t)}{\mathrm{d}t} = -(1 - P_C(t)) \iint\limits_{S(C_t)} f^{B_t}_{\mathbf{R}(t)}(\mathbf{p}) \int_{-\infty}^{w_n(\mathbf{p},t)} f^{B_t}_{V_n(t)}\left(v_n | D_t^{\mathbf{p}}\right) v_n \, \mathrm{d}v_n \, \mathrm{d}S, \tag{4.19}$$

where the effects of the rotation of the surface elements disappear as $\|P_S\| \to 0$ and the smooth element resizing does not affect the instantaneous probability flow.

Equation 4.19 is defined for the three-dimensional case where the conflict region is a volume and the conflict region boundary is a piece-wise smooth surface. When the derivation of Equation 4.19 is repeated for the two-dimensional case where the conflict region C_t is an area in the plane and the conflict region boundary is a piece-wise smooth curve $C(C_t)$, the probability flow is given by

$$\frac{\mathrm{d}P_C(t)}{\mathrm{d}t} = -(1 - P_C(t)) \int\limits_{C(C_t)} f^{B_t}_{\mathbf{R}(t)}(\mathbf{p}) \int_{-\infty}^{w_n(\mathbf{p},t)} f^{B_t}_{V_n(t)}\left(v_n | D_t^{\mathbf{p}}\right) v_n \, \mathrm{d}v_n \, \mathrm{d}s, \tag{4.20}$$

where the integration parameter s is the arc length along the conflict area boundary.

Having reformulated the probabilistic conflict detection problem using probability flow, we present two simplifications in the next two sections which enable the computation of the conflict probability in real-time: defining a tight upper bound to the conflict probability and assuming Gaussian distribution of the vehicle states.

4.3 Derivation of Tight Upper Bound on the Conflict Probability

The first simplification to the formulation of the probability flow in Equations 4.19 and 4.20 is to assume that the probability density function associated with the vehicle states stays unaffected by previous conflict throughout the prediction period. This assumption induces an upper bound on the probability flow which induces an upper bound to the conflict probability according to

Equation 4.1. It also simplifies the propagation of the probability density functions associated with the vehicle states.

In order to calculate an upper bound to the probability flow given in Equations 4.19 and 4.20, we define the set of trajectories located outside the conflict region at time t as

$$H_t \triangleq \{\omega \in \Omega : \mathbf{R}(t,\omega) \notin C_t\}. \tag{4.21}$$

When we compare H_t to B_t from Equation 3.2, we see that $B_t \subseteq H_t$ and we construct the upper bound to Equation 4.5 as follows:

$$\begin{aligned} \frac{\mathrm{d}P_C(t)}{\mathrm{d}t} &= \lim_{\Delta t \to 0} \left(\frac{1}{\Delta t} P\left[\Delta A_{t:t+\Delta t} \cap B_t\right] \right) \\ &\leq \lim_{\Delta t \to 0} \left(\frac{1}{\Delta t} P\left[\Delta A_{t:t+\Delta t} \cap H_t\right] \right) \triangleq \frac{\mathrm{d}P_C^{UB}(t)}{\mathrm{d}t}. \end{aligned} \tag{4.22}$$

In Equation 4.5, we calculate the probability flow due to trajectories that have not earlier entered the conflict volume, whereas for the upper bound calculation, we calculate the probability flow due to all trajectories in the sample space Ω, whether they have earlier entered the conflict volume or not. This upper bound has also been used by Jones [37, 38] to reduce computational complexity. When we repeat the derivation for the three-dimensional case of Equation 4.19 for the upper bound definition, we obtain the following:

$$\frac{\mathrm{d}P_C^{UB}(t)}{\mathrm{d}t} = -\iint\limits_{S(C_t)} f_{\mathbf{R}(t)}(\mathbf{p}) \int_{-\infty}^{w_n(\mathbf{p},t)} f_{V_n(t)}\left(v_n | D_t^{\mathbf{p}}\right) v_n \,\mathrm{d}v_n \,\mathrm{d}S. \tag{4.23}$$

The derivation for the two-dimensional case of Equation 4.20 for the upper bound definition yields

$$\frac{\mathrm{d}P_C^{UB}(t)}{\mathrm{d}t} = -\int\limits_{C(C_t)} f_{\mathbf{R}(t)}(\mathbf{p}) \int_{-\infty}^{w_n(\mathbf{p},t)} f_{V_n(t)}\left(v_n | D_t^{\mathbf{p}}\right) v_n \,\mathrm{d}v_n \,\mathrm{d}s. \tag{4.24}$$

Note that Equations 4.23 and 4.24 only use probability density functions unaffected by previous conflict. This is advantageous because the process of calculating the propagation of probability density functions distorted by conflict is computationally expensive.

The upper bound to the probability of conflict can now be calculated by replacing $\frac{\mathrm{d}P_C(t)}{\mathrm{d}t}$ in Equation 4.1 with $\frac{\mathrm{d}P_C^{UB}(t)}{\mathrm{d}t}$, yielding

$$P_C^{UB}(t_f) = \int_{t_0}^{t_f} \frac{\mathrm{d}P_C^{UB}(\tau)}{\mathrm{d}\tau} \,\mathrm{d}\tau. \tag{4.25}$$

The calculation of an upper bound may seem undesirable, but it is a tight bound at or below typical threshold values. This can be understood by considering that the conflict probability threshold below which the vehicle planned path is considered safe is usually chosen quite low. For safe vehicle planned paths, the probability that a vehicle trajectory will enter the conflict region is low, therefore the probability that a vehicle trajectory will enter the conflict zone more than once in the prediction period can reasonably expected to be very low. According to this reasoning, H_t is a good approximation of B_t and the upper bound is therefore quite tight. For values of the probability of conflict above the threshold, an upper bound that differs much from the actual conflict probability would not impact on the operation of the conflict avoidance system. A conflict avoidance system using the upper bound definition for conflict detection will therefore be cautious, but not overly so. The examples in Chapter 5 confirm the tight upper bound for conflict probability values below typical threshold values.

4.4 Gaussian Vehicle States Approximation

The second simplification to the definition of probability flow of Equations 4.19 and 4.20 is to assume that the probability density function associated with the vehicle states has a jointly Gaussian distribution.

Suppose the joint probability distribution of the vehicle position and velocity states can be described by a Gaussian distribution, or

$$\begin{bmatrix} \mathbf{V}(t,\omega) \\ \mathbf{R}(t,\omega) \end{bmatrix} \sim N\left(\begin{bmatrix} \overline{\mathbf{V}}(t) \\ \overline{\mathbf{R}}(t) \end{bmatrix}, \begin{bmatrix} \mathbf{C_V}(t,t) & \mathbf{C_{VR}}(t,t) \\ \mathbf{C_{VR}}^T(t,t) & \mathbf{C_R}(t,t) \end{bmatrix} \right). \tag{4.26}$$

This is a common assumption in probabilistic conflict detection methods [34, 37, 38, 55–62] because vehicle and environment state estimators are usually assumed to provide normally distributed state estimates. The motivation for this assumption for autonomous underwater vehicles is given in Section 2.4.

The conditional distribution of $\mathbf{V}(t)$ given the vehicle position $\mathbf{p}$ is stated as [12]

$$\begin{aligned} \mathbf{V}(t,\omega)|D_t^{\mathbf{p}} \sim N\big(& \overline{\mathbf{V}} + \mathbf{C_{VR}}(t,t)\mathbf{C_R}^{-1}(t,t)\left(\mathbf{p} - \overline{\mathbf{R}}(t)\right), \\ & \mathbf{C_V}(t,t) - \mathbf{C_{VR}}(t,t)\mathbf{C_R}^{-1}(t,t)\mathbf{C_{VR}}^T(t,t)\big). \end{aligned} \tag{4.27}$$

The conditional distribution of the normal velocity $V_n(t,\omega) = \mathbf{V}(t,\omega)\cdot\mathbf{n}$ given the vehicle position $\mathbf{p}$ is also Gaussian and given by [12]

$$V_n(t,\omega)|D_t^{\mathbf{p}} \sim N\left(\overline{V}_n(t), \sigma_V^2(t)\right), \tag{4.28}$$

where

$$\overline{V}_n(t) = \mathbf{n}^T\left(\overline{\mathbf{V}}(t) + \mathbf{C_{VR}}(t,t)\mathbf{C_R}^{-1}(t,t)\left(\mathbf{p} - \overline{\mathbf{R}}(t)\right)\right) \tag{4.29}$$

and

$$\sigma_V^2(t) = \mathbf{n}^T\left(\mathbf{C_V}(t,t) - \mathbf{C_{VR}}(t,t)\mathbf{C_R}^{-1}(t,t)\mathbf{C_{VR}}^T(t,t)\right)\mathbf{n}. \tag{4.30}$$

The integral over v_n in Equations 4.23 and 4.24 can be calculated as follows:

$$\begin{aligned} \int_{-\infty}^{w_n(\mathbf{p},t)} f_{V_n(t)}\left(v_n|D_t^{\mathbf{p}}\right) v_n \,\mathrm{d}v_n = & \int_{-\infty}^{w_n(\mathbf{p},t)} \varphi_{\overline{V}_n(t),\sigma_V^2(t)}(v_n)\left(v_n - \overline{V}_n(t)\right)\,\mathrm{d}v_n \\ & + \overline{V}_n(t)\int_{-\infty}^{w_n(\mathbf{p},t)} \varphi_{\overline{V}_n(t),\sigma_V^2(t)}(v_n)\,\mathrm{d}v_n \\ = & -\frac{\sigma_V(t)}{\sqrt{2\pi}}\exp\left(-\frac{\left(w_n(\mathbf{p},t) - \overline{V}_n(t)\right)^2}{2\sigma_V^2(t)}\right) \\ & + \overline{V}_n(t)\Phi_{\overline{V}_n(t),\sigma_V^2(t)}\left(w_n(\mathbf{p},t)\right), \end{aligned} \tag{4.31}$$

where $\varphi_{\mu,\sigma^2}(\cdot)$ is the Gaussian probability density function and $\Phi_{\mu,\sigma^2}(\cdot)$ is the Gaussian cumulative distribution function, both with mean μ and variance σ^2. The Gaussian cumulative distribution function in Equation 4.31 can be read from a table of the normalized Gaussian cumulative distribution function $\Phi(\cdot)$ or efficiently calculated using the complementary error function $\mathrm{erfc}(\cdot)$ as follows:

$$\begin{aligned} \Phi_{\overline{V}_n(t),\sigma_V^2(t)}\left(w_n(\mathbf{p},t)\right) &= \Phi\left(\frac{w_n(\mathbf{p},t) - \overline{V}_n(t)}{\sigma_V(t)}\right) \\ &= \frac{1}{2}\mathrm{erfc}\left(\frac{\overline{V}_n(t) - w_n(\mathbf{p},t)}{\sqrt{2}\sigma_V(t)}\right). \end{aligned} \tag{4.32}$$

The advantages of using the assumption of Gaussian distributed states are that the inner integral of Equations 4.23 and 4.24 is calculated easily and that the vehicle and environment states are simple to propagate into the future.

4.5 Adaptive Integration

The assumptions of Sections 4.3 and 4.4 simplifies the calculation of the probability of conflict by making the vehicle and environment state propagation efficient as well as enabling the inner integral of Equations 4.23 and 4.24 to be calculated efficiently. The last step in constructing a method to calculate the probability of conflict in real-time is to find an efficient method for numerical integration.

To calculate the upper bound to the probability of conflict, $P_C^{UB}(t_f)$, we need to integrate along the boundary of the conflict region as given by Equation 4.23 or 4.24, and then integrate over the time interval $[t_0, t_f]$ as given by Equation 4.25. This may seem difficult to calculate in real-time. However, when considering that the probability flow is typically concentrated on only a small section of the conflict region boundary at any given time, an adaptive numerical integration method can be used to efficiently calculate both the surface or line integral and the time integral.

The central concept of adaptive integration [14, 17] is to initially partition the integration domain quite roughly and only refine the integration estimate in those sections of the integration domain where the estimated error is large. The integration estimate for a function f is obtained from an integration rule Qf and the error estimate is obtained from the error rule Ef. Although the integration and error rules may differ, the basic adaptive integration procedure stays the same and is shown in recursive formulation as pseudocode in Algorithm 1[4].

Algorithm 1 Recursive adaptive integration

function AdaptiveIntegration(domain)

```
1: Divide integration domain into smaller sections
2: for each section do
3:    Evaluate integration rule Qf
4:    Evaluate error estimate Ef
5:    if |Ef| > error bound then
6:       AdaptiveIntegration(section)
7:    else
8:       Add Qf to integration result
9:    end if
10: end for
```

By choosing the initial partition of the integration domain quite rough and only refining those sections where the estimated error is large, the computation time is brief due to the fact that in the calculation of the probability of conflict, the fraction of sections in the integration domain to be refined is typically limited.

The field of numerical integration is relatively mature for low-dimensional integrals and a multitude of relevant options exists for the integration rule Qf [14, 17, 45]. There is a trade-off between the complexity of the integration rule and the execution time of the integration rule: a more complex rule provides a more accurate integration estimate which requires fewer refinement levels, but the evaluation of the integration rule over the initial partition executes in a longer time mainly due to more integrand evaluations per rule. We opt for fairly simple integration rules which provide fast evaluation in the sections of the integration domain with low probability flow, but which also require more refinement levels in the sections with high probability flow. Our choices for integration and error rules for the one-dimensional integrals and surface integrals are detailed in the next two subsections.

[4]The number of refinement levels (recursion depth) is often restricted to a predefined maximum.

4.5.1 Calculating One-dimensional Integrals

Calculating the upper bound to the probability of conflict requires calculating two one-dimensional integrals: the line integral of Equation 4.24 for the two-dimensional problem formulation and the integration of the probability flow over the prediction time period in Equation 4.25 for both the two- and three-dimensional problem formulations.

The mean and covariance of the vehicle states are typically available at equally-spaced time instances. The use of one of the Newton-Cotes rules [14, 17, 45] therefore becomes imperative for the integration over time of Equation 4.25. For the sake of simplicity, we use the same integration scheme for the line integration of Equation 4.24. We choose Simpson's rule [45], which is defined for the integrand $f(\cdot)$ and interval $[a, b]$ as

$$Q^S_{[a,b]} f \triangleq \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]. \tag{4.33}$$

Simpson's rule calculates the integral of a quadratic polynomial fit through three equally-spaced points. The error between the actual integral and Simpson's rule is given by

$$\int_a^b f(x)\,\mathrm{d}x - Q^S_{[a,b]} f = -\frac{(b-a)^5}{2880} f^{(4)}(\xi) \tag{4.34}$$

where $\xi \in [a, b]$. Simpson's rule is therefore exact for polynomials of degree three or less.

When integrating adaptively using Simpson's rule, we first partition the integration domain into intervals with equal length. Let the interval $[a, b]$ denote one of these intervals, shown in Figure 4.3. We then find the midpoint of the interval using $c = \frac{1}{2}(a+b)$ and evaluate Simpson's

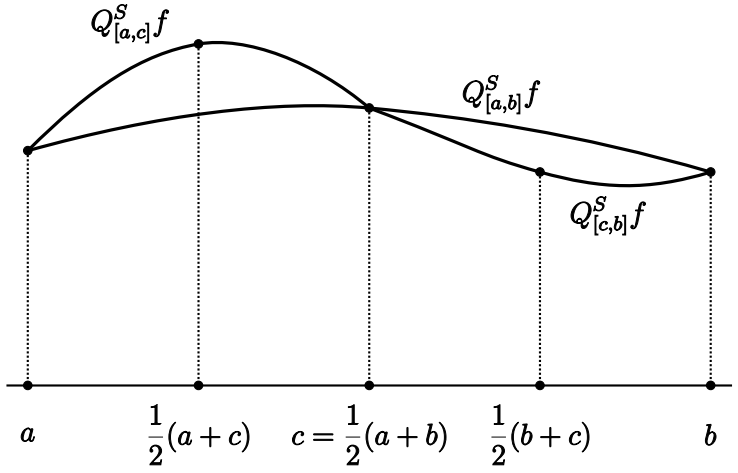


Figure 4.3: Diagram of adaptive integration using Simpson's rule

rule $Q^S_{[a,b]} f$. To obtain an error estimate, we evaluate Simpson's rule over both halves of the interval – $Q^S_{[a,c]} f$ and $Q^S_{[c,b]} f$ respectively – and use the error estimate

$$E^S_{[a,b]} f = \left| Q^S_{[a,b]} f - Q^S_{[a,c]} f - Q^S_{[c,b]} f \right|. \tag{4.35}$$

The sum of the integration rules over the two halves of the interval, $Q^S_{[a,c]} f + Q^S_{[c,b]} f$, is taken as the best estimate of the actual integral. When the error estimate exceeds a certain threshold, the above procedure is repeated for each half of the interval $[a, b]$. This refinement of the integration estimate is repeated until the error estimate is lower than the error threshold or a certain

minimum interval length is reached. We use a relative error threshold which is proportional to the interval length.

The choice of the length of the intervals in the initial partition of the integration domain is a trade-off between execution speed and accuracy of the error estimate: a longer interval will cause a faster execution time, whereas the interval needs to be short enough so that the error estimate of Equation 4.35 at worst underestimates the size of the actual error by a small margin. See Appendix A for an analysis of this requirement.

4.5.2 Calculating Surface Integrals

For the surface integral of Equation 4.23, we assume that the boundary of the conflict volume is a triangulated surface. It is common to represent the surface of a general volume with a mesh of triangles in practice, hence the assumption.

The integration rule chosen for the surface integration is the simplest possible, namely the midpoint rule. The midpoint rule approximates the surface integral by multiplying the area of the triangle by the value of the integrand at the centroid of the triangle, or

$$Q^M_{(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3)} f = A\left(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3\right) f\left(\frac{1}{3}\left(\mathbf{p}_1+\mathbf{p}_2+\mathbf{p}_3\right)\right), \tag{4.36}$$

where $\mathbf{p}_1$, $\mathbf{p}_2$ and $\mathbf{p}_3$ are the vertices of the triangle as shown in Figure 4.4 and $A\left(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3\right)$

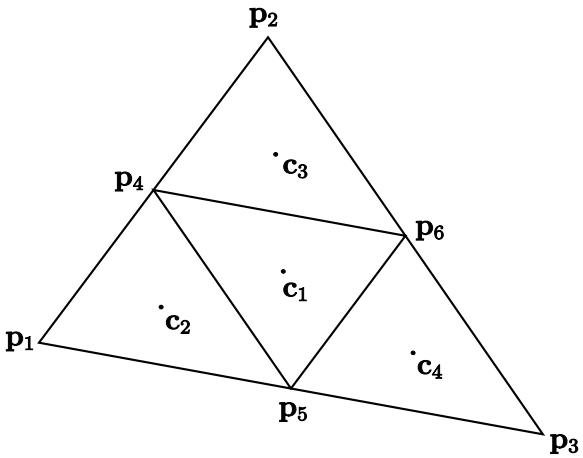


Figure 4.4: Diagram of triangle used for adaptive surface integration

denotes the area of the triangle. Similar to the midpoint rule for one-dimensional integration, the midpoint rule for surface integrals is exact for a multivariate polynomial that is linear in its variables.

To obtain an error estimate used in the adaptive integration procedure, we divide the triangle into four congruent triangles as shown in Figure 4.4 with $\mathbf{p}_4 = \frac{1}{2}(\mathbf{p}_1+\mathbf{p}_2)$, $\mathbf{p}_5 = \frac{1}{2}(\mathbf{p}_1+\mathbf{p}_3)$ and $\mathbf{p}_6 = \frac{1}{2}(\mathbf{p}_2+\mathbf{p}_3)$. We then apply the integration rule to each of the smaller triangles and estimate the error by calculating the difference between the integration rule on the large triangle and the sum of the integration rule on each of the smaller triangles, or

$$E^M_{(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3)} f = \left|\frac{3}{4} Q^M_{(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3)} f - Q^M_{(\mathbf{p}_1,\mathbf{p}_4,\mathbf{p}_5)} f - Q^M_{(\mathbf{p}_5,\mathbf{p}_6,\mathbf{p}_3)} f - Q^M_{(\mathbf{p}_4,\mathbf{p}_2,\mathbf{p}_6)} f\right|, \tag{4.37}$$

where we recognise that $Q^M_{(\mathbf{p}_6,\mathbf{p}_5,\mathbf{p}_4)} f = \frac{1}{4} Q^M_{(\mathbf{p}_1,\mathbf{p}_2,\mathbf{p}_3)} f$. The sum of the integration rule on each of the smaller triangles is taken as the best approximation to the actual integral. When the

error estimate of Equation 4.37 exceeds a certain error threshold, each of the smaller triangles is divided into four congruent triangles. The integration rule is then applied to each of these triangles, the error estimate calculated and the process repeated until the error estimate is lower than the error threshold or until a minimum triangle size is reached.

The size of the triangles in the surface mesh is constrained by the smallest singular value of the covariance matrix of the probability density function of the vehicle position. This is due to the definition of the error estimate – narrow peaks in the integrand of the surface integral should not be overlooked. The error estimate of Equation 4.37 should therefore be analysed for each problem formulation to ensure at worst a small underestimation of the integration error.

This subsection concludes the description of the simplifications and numerical techniques necessary to calculate an upper bound to the probability of conflict in real-time for both the two-dimensional and three-dimensional problem descriptions. The next chapter describes two examples that elucidate these techniques and show that the conflict probability can indeed be calculated in real-time.

Chapter 5

Conflict Detection Examples

This last chapter in Part I illustrates the method developed in Chapter 4 for the efficient calculation of the probability of conflict, using simulation examples. This chapter presents two examples, each of which has been designed to illustrate different concepts. The first example, described in Section 5.1, uses a well-known two-dimensional example problem that is often found in the conflict detection literature and serves to compare the performance of the method employed in this study with that of some of the existing conflict detection methods. The second example, described in Section 5.2, is a three-dimensional example which illustrates the capabilities of our method.

The following two sections have partly been adapted from a paper recently published by the author [76].

5.1 Two Airplanes Example

The first example compares the performance of the probability flow method with other widely-used methods. The selected example problem was introduced by Paielli and Erzberger [55] – that of two airplanes with crossing flight paths. Yang et al. [84] and Jones [37] also used this problem in their examples.

5.1.1 Problem Description

This example is a two-dimensional problem where the height information is disregarded. The separation distance between the two airplanes should always exceed 5 nmi, therefore the conflict region is given by a circle with a 5 nmi radius. The sizes of the airplanes are small compared to the conflict area and the airplanes are therefore approximated as point masses. The mean velocity of each airplane is assumed to be constant during the encounter. The uncertainty in the position and velocity of each airplane is modelled as a joint Guassian distribution. Each airplane's cross-track position standard deviation is 0.5 nmi and its along-track position standard deviation grows at a rate of 0.25 nmi/min. A conceptual diagram of the example is shown in Figure 5.1. One airplane is chosen as the reference (R) while the other is called the intruder (I). For this example, the flight paths cross at a 90 ° angle and the mean paths are straight for the duration of the encounter. We compare the probability flow method with the analytic method of Paielli and Erzberger [55] and the geometric Monte Carlo method of Yang et al. [84].

5.1.2 Implementation

We model the airplane dynamics by a linear time-invariant four-state model. This time-invariant model is a special case of the linear state space model discussed in Section 2.1. The dynamic

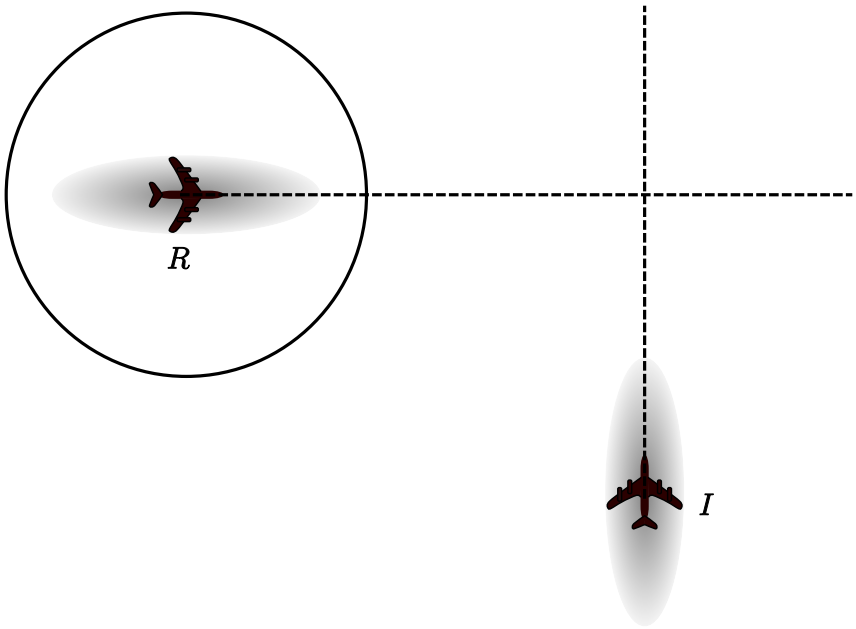


Figure 5.1: Two airplanes example (not to scale)

vehicle model enables us to propagate the mean and covariance of the joint distribution of the airplane position and velocity throughout the prediction period in question.

We use the probability flow method developed in Chapter 4 to calculate a tight upper bound to the probability of conflict for vehicle states with a Gaussian distribution. In short, this calculation involves numerical integration of an iterated integral: the inner integral calculates an upper bound to the probability flow as given by Equation 4.24 and the outer integral calculates an upper bound to the probability of conflict as given by Equation 4.25.

For clarity's sake, the different components of the probability flow method developed in Chapter 4 as applied to this example are summarised in the following paragraphs.

Obtain the current mean and covariance. In practice we would obtain the mean and covariance of each airplane's current position and velocity from the state estimator associated with each airplane. However, for this example, we assign an initial distribution with a cross-track and along-track position standard deviation of 0.5 nmi to each airplane.

Partition the prediction period. Next, we partition the prediction time period into equal intervals. This is the initial rough partition that forms the first step in the adaptive integration of Equation 4.25. In this case, the prediction period is 10 minutes and the length of the intervals in the initial partition is is 1 minute.

Propagate and transform the mean and covariance. For each interval in the partition of the prediction period, we initially propagate the mean and covariance of each airplane's position and velocity to the start, middle and end of the interval, using the equations of Section 2.1. If the adaptive integration procedure refines the integration estimate for a specific interval, the mean and covariance are propagated to the additional required time instants. After the means and covariances of the two airplanes are propagated, they are combined using the transformation in Subsection 3.2.2.

Adaptively integrate over the prediction period. We solve Equation 4.25 by integrating adaptively (Algorithm 1) using Simpson's rule for each of the intervals in the partition of the prediction period as set out in Subsection 4.5.1. The adaptive integration procedure is restricted to 4 refinement levels.

Adaptively integrate along the conflict volume boundary. We calculate the integrand of Equation 4.25 by solving Equation 4.24 at each time instant. We solve Equation 4.24 by adaptive integration (Algorithm 1) along the circle forming the boundary of the conflict area, using Simpson's rule as set out in Subsection 4.5.1. The conflict area boundary is partitioned into 10 equal-sized arcs at the start of the adaptive integration procedure.

Calculate the integrand of the line integration. The inner integral of Equation 4.24 is efficiently calculated using Equations 4.29 through 4.32 with $w_n(\mathbf{p}, t)$ set to zero and using the combined propagated mean and covariance at the time instant t. $f_{\mathbf{R}(t)}(\mathbf{p})$ in Equation 4.24 is calculated from the multivariate Gaussian probability density function using the combined propagated mean and covariance at the time instant t.

This simulation example of the probability flow method was implemented in the Python programming language with the bottleneck calculations implemented in the C programming language. For the sake of comparison, the analytic method of Paielli and Erzberger[55], the geometric Monte Carlo method of Yang et al.[84] and a full Monte Carlo simulation were applied to the same example problem. The latter three methods were implemented in Matlab.

5.1.3 Results

The results of the simulations are shown in Table 5.1[1]. The full Monte Carlo result is taken as

Table 5.1: Results of two airplanes example

Method	P_C	Error[a](%)	Uncer-tainty[b]	Execution time (s)
Probability flow	0.10452	2.6	0.00016	0.002
Geometric Monte Carlo	0.09396	-7.8	0.00087	0.452
Paielli and Erzberger's method	0.09412	-7.6	-[c]	-[d]
Monte Carlo	0.10188	-	0.00030	47915

[a]Compared to Monte Carlo simulation.
[b]Estimated error bound for probability flow; 3× standard deviation for Monte Carlo methods, which is calculated using Equation B.17.
[c]The method of Paielli and Erzberger provides no way to calculate the uncertainty in the answer.
[d]The method of Paielli and Erzberger is an analytic method and executes approximately instantaneously.

the true probability of conflict, therefore, the error is calculated as the deviation from the full Monte Carlo simulation. The probability flow method slightly overestimates the probability of conflict (confirming that it calculates a tight upper bound) compared to larger underestimates by the other methods. We expect the uncertainty in the result of the probability flow method – calculated by the accumulating the error rule of the adaptive integration procedure as set out in Subsection 4.5.1 – to be an overestimation of the actual error and we therefore compare it to three times the standard deviation for the Monte Carlo methods. It should be noted that the execution times should not be compared directly because the method implementations differ. The execution times only provide a rough indication of the efficiency of the methods. The probability flow method execution time of 2 ms confirms that it can calculate the conflict probability in real-time.

These results show that the probability flow method is very efficient and compares well to other methods for simple vehicle dynamics and conflict regions. It should be noted that the

[1]The method implementations were run on a desktop computer with an Intel dual-core 2.2 GHz processor and 2 GB RAM.

probability flow method and Monte Carlo methods can manage more complex vehicle dynamics and more general conflict regions, but the problem formulation is kept simple in order to compare the methods to that of Paielli and Erzberger, which contains limiting assumptions.

5.2 Autonomous Underwater Vehicle Example

The second example illustrates the performance of the probability flow method for a complex vehicle maneuver in a cluttered environment. The result of this simulation is compared to the result of a full Monte Carlo simulation.

5.2.1 Problem Description

This example is a three-dimensional problem and consists of an autonomous underwater vehicle (AUV) in a harbour environment, shown in Figure 5.2.

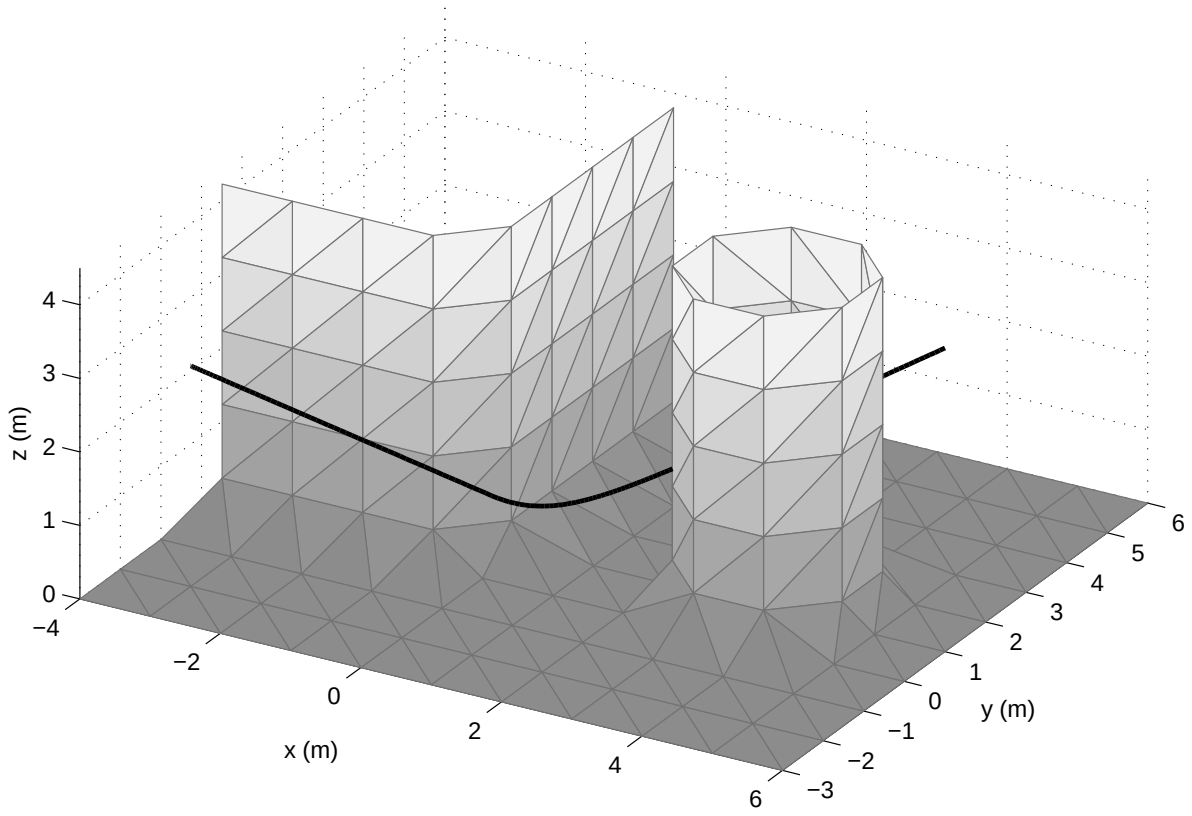


Figure 5.2: AUV example environment and trajectory

The probability distribution of the vehicle states is assumed to be Gaussian. This assumption is motivated for AUVs in Section 2.4 and allows us to use the probability flow method developed in Chapter 4.

The conflict volume already incorporates both the vehicle and obstacle volumes, is time-invariant and its surface is approximated by a mesh of 258 triangles. In practice, the conflict volume surface would automatically be reconstructed from a set of sample points. There are a number of algorithms available to do this (e.g. [4, 31]) – but for this example the conflict volume surface is constructed beforehand. This surface reconstruction process introduces another source of errors, but if the approximated surface completely encloses the true conflict volume, the

calculated probability of conflict is an upper bound to the probability of conflict calculated on the true surface. For this example, the triangulated surface is taken as the true surface. The maximum triangle size is restricted by the minimum singular value of the vehicle position covariance matrix – this ensures that a spot of high probability flow through the conflict volume surface is not missed by the numerical integration algorithm. See Appendix A for an analysis of this restriction.

The length of the prediction period for this example is 2 minutes 40 seconds. The conflict volume surface and mean vehicle trajectory is shown in Figure 5.2.

5.2.2 Implementation

The dynamics of the AUV is modelled by a six-state time-invariant linear model. This time-invariant model is a special case of the linear state space model described in Section 2.1. The dynamic model enables us to propagate the mean and covariance of the joint distribution of the vehicle position and velocity throughout the length of the prediction period.

For this example, we assume that the transformation that combines the volumes and probability distributions of the vehicle and obstacles as described in Subsection 3.2.2 has already been computed. The example setup therefore consists of a point mass with uncertain states and a conflict volume without uncertainty.

We calculate a tight upper bound to the probability of conflict for normally distributed vehicle states using the probability flow method developed in Chapter 4. In short, the method calls for numerical integration of an iterated integral: the inner integral is a surface integral which calculates an upper bound to the probability flow as given by Equation 4.23 and the outer integral calculates an upper bound to the probability of conflict as given by Equation 4.25.

For clarity's sake, we summarise the different components of the probability flow method developed in Chapter 4 as applied to this example in the following paragraphs.

Obtain the current mean and covariance. In practice we would obtain the mean and covariance of the vehicle position and velocity from the state estimator. However, for this example, we assign an arbitrary distribution to the initial mean and covariance.

Partition the prediction period. Next, we partition the prediction time period into equal intervals. This is the initial rough partition that forms the first step in the adaptive integration of Equation 4.25. In this case, the prediction period is 2 minutes and 40 seconds and the length of the intervals in the initial partition is 16 seconds.

Propagate the mean and covariance. For each interval in the partition of the prediction period, we initially propagate the mean and covariance of the vehicle's position and velocity to the start, middle and end of the interval, using the equations of Section 2.1. If the adaptive integration procedure refines the integration estimate for a specific interval, the mean and covariance are propagated to the additional required time instants.

Adaptively integrate over the prediction period. We solve Equation 4.25 by integrating adaptively (Algorithm 1) using Simpson's rule for each of the intervals in the partition of the prediction period as set out in Subsection 4.5.1. The adaptive integration procedure is restricted to 4 refinement levels.

Adaptively integrate along the conflict volume boundary. We calculate the integrand of Equation 4.25 by solving Equation 4.23 at each time instant. We solve Equation 4.23 by adaptive integration (Algorithm 1) over the triangulated surface forming the boundary of the conflict volume, using the midpoint rule as set out in Subsection 4.5.2.

Calculate the integrand of the line integration. The inner integral of Equation 4.23 is efficiently calculated using Equations 4.29 through 4.32 with $w_n(\mathbf{p}, t)$ set to zero and using the combined propagated mean and covariance at the time instant t. $f_{\mathbf{R}(t)}(\mathbf{p})$ in Equation 4.23 is calculated from the multivariate Gaussian probability density function using the combined propagated mean and covariance at the time instant t.

The probability flow method was implemented in Python with the bottleneck computations implemented in C. The results are compared to a full Monte Carlo simulation which was implemented in Matlab[2].

5.2.3 Results

The results of the probability flow method and Monte Carlo simulation are shown in Table 5.2[3]. The error is calculated as the deviation from the baseline Monte Carlo simulation in

Table 5.2: Results of AUV example

Method	P_C	Error[a](%)	Uncertainty[b]	Execution time (s)
Probability flow	0.09265	7.8	0.00068	0.094
Monte Carlo	0.08592	-	0.00060	143681

[a]Compared to Monte Carlo simulation.

[b]Estimated error bound for probability flow; 3× standard deviation for Monte Carlo method, which is calculated using Equation B.17.

percentage. The probability flow method overestimates the probability of conflict as calculated by the Monte Carlo method by 7.8%, confirming that the probability flow method calculates a tight upper bound to the probability of conflict. The error in the conflict probability as calculated by the probability flow method is slightly higher than in the first example because the period during which the vehicle is predicted to be in the vicinity of the conflict volume is almost as long as the total prediction period. The probability that a trajectory enters the conflict volume multiple times during the prediction period is higher than for a short conflict period, and the upper bound is therefore slightly higher (see the reasoning in Section 4.3). The uncertainty in the result of the probability flow method is calculated by accumulating the error estimates of the adaptive integration procedure as set out in Section 4.5. This is expected to be an overestimation of the size of the actual error. Note that the execution times should not be compared directly because the method implementations differ – they only serve to provide a rough indication of the method efficiency. The execution time of the probability flow method is less than 100 ms, which confirms that the conflict probability upper bound can be computed in real-time.

This second example shows that the probability flow method can calculate a tight upper bound to the probability of conflict with Gaussian distributed vehicle states in real-time, even for complex[4] vehicle maneuvers and cluttered conflict environments. To the best of the author's knowledge, only the Monte Carlo simulation method can manage such a general problem

[2]The geometric Monte Carlo method of Yang et al. [84] becomes indistinguishable from the standard Monte Carlo method for complex maneuvers and is therefore omitted.

[3]The method implementations were run on a desktop computer with an Intel dual-core 2.2 GHz processor and 2 GB RAM.

[4]As opposed to simple trajectories such as straight lines.

formulation, but it cannot calculate the probability of conflict in real-time using the example implementation[5]. We therefore conclude that the probability flow method is a significant contribution to the field of probabilistic conflict detection.

[5]We recognise that there may be more efficient Monte Carlo techniques (such as the RESTART method by Villén-Altamirano et al. [78]). However, the focus of the examples was on comparing the probability flow method to *existing* Monte Carlo conflict detection methods.

Part II

Conflict Resolution

Chapter 6

Overview of Conflict Resolution

The focus of Part I is on finding an efficient method to calculate the probability of conflict. Part II deals with the problem of conflict resolution which entails the design of a safe alternative path in a situation where the probability of conflict exceeds the threshold for safe operation. This chapter introduces Part II by reviewing the different approaches to conflict resolution and introducing important concepts which lays the foundation for the design of an efficient method in Chapter 7.

This chapter discusses the two main approaches to conflict resolution (Section 6.1), then introduces important planning concepts (Section 6.2) after which the conflict resolution problem is formally defined using the replanning approach (Section 6.3). Section 6.4 reviews the existing planning methods that can be used for conflict resolution.

6.1 Approaches to Conflict Resolution Methods

Once we have established a method to test a section of the planned path for conflict, the question arises of how to resolve any conflict detected along the planned path. The formulation of this conflict resolution problem differs widely among researchers, but the existing approaches can be divided into two groups: the *single avoidance maneuver approach* and the *replanning approach*. These two approaches are summarised in the following subsections after which the focus falls on the replanning approach.

6.1.1 Single Avoidance Maneuver Approach

The conflict resolution methods using this approach attempt to choose or design a single maneuver to avoid the conflict. These methods do no replanning and the task of reaching the goal state or next waypoint is assumed to be managed by a pilot or another subsystem. Most of the conflict resolution methods surveyed by Kuchar and Yang [43] fall into this category. The simplest of these have only one avoidance maneuver choice, for example in a ground proximity warning system (GPWS) for airplanes where the only available avoidance maneuver is to pull up. More sophisticated methods choose from a set of available avoidance maneuvers. The maneuver choice is made according to a set of rules or the optimisation of a cost function. In addition to choosing *which* avoidance maneuver to execute, these methods have to decide *when* to execute the maneuver. An important factor in making these decisions is the trade-off between the probability of conflict and the probability of a false alarm. This trade-off is visualised by the System Operating Characteristic (SOC) curve which was developed by Kuchar [42].

Several of the methods using the single avoidance maneuver approach have been designed for air traffic control (ATC) application. In such an application, the conflict resolution subsystem is part of an alerting system, advising the pilot of a suitable avoidance maneuver should any conflict be detected. The pilot then has to steer the airplane back to the planned path when

the conflict situation has been negotiated. For fully autonomous vehicles using this approach, the task of rejoining the planned path has to be managed by an online path planner. The newly-planned path should then be tested for conflict by the conflict detection subsystem. In cluttered environments, it is not inconceivable that this interaction between the conflict detection subsystem and path planner could deadlock when the newly-planned path is repeatedly rejected, which could result in conflict. A solution would be to combine the conflict detection subsystem and path planner, which is precisely the replanning approach (Subsection 6.1.2). The single avoidance maneuver approach for fully autonomous vehicles is therefore only applicable to uncluttered environments where the vehicle is unlikely to encounter a second conflict situation as a result of resolving a conflict situation.

Despite the simplicity of these methods, their failure to function in cluttered environments or guarantee that a goal state can be reached cause us to reject them for application to autonomous vehicles – a replanning approach is therefore preferable.

6.1.2 Replanning Approach

The methods using the replanning approach attempt to find a path from an initial state to a goal state while satisfying both the differential and global[1] constraints. In addition, they often attempt to optimise a cost function associated with the vehicle path. This *motion planning* problem is studied in slightly different forms in three fields, namely robotics, artificial intelligence and control theory. If differential constraints are neglected, then there are two distinct ways in which this problem is approached: *combinatorial* motion planning and *sampling-based* motion planning [48]. *Combinatorial motion planning* algorithms explicitly construct the conflict space and attempt to find a trajectory in the free space. They are exact and complete – if a feasible path exists, these algorithms are guaranteed to find a solution. *Sampling-based motion planning* algorithms avoid the explicit construction of the conflict space by exploring the free space by means of a sampling scheme, which means that their only interaction with the conflict space model is through the conflict test of a path section. These algorithms are neither exact nor complete – they cannot guarantee finding a feasible path if one exists.

When differential constraints are added, the combinatorial approach becomes intractable [48], while the sampling-based approach produces good results despite its lack of completeness. Motion planning with second-order or higher differential constraints is called *kinodynamic planning*. From here on we view the conflict resolution problem for autonomous vehicles as motion planning under global constraints and second-order or higher differential constraints and we follow the sampling-based approach to find a feasible path – we therefore use the term *sampling-based kinodynamic planning*.

6.2 Sampling-based Planning Concepts

As mentioned in Section 6.1, we choose the sampling-based kinodynamic planning approach for conflict resolution. Informally, this approach involves finding a vehicle input that would steer the vehicle from the initial state to the goal state while satisfying both differential and global constraints, by means of sampling-based search techniques. In this section, we introduce some important concepts necessary to understand sampling-based planning. This is followed by a formal problem definition (Section 6.3) and a review of the existing methods (Section 6.4).

[1] In most texts, satisfying global constraints means avoiding collisions. In this thesis, a path satisfying the global constraints means that the probability of conflict associated with the path is below the threshold for safe operation.

6.2.1 Sampling Space

All sampling-based planning methods sample a space of vehicle states – either directly or indirectly – and attempt to connect these states by feasible[2] path sections with the goal of finding a feasible path between the initial and goal states. Researchers have used different formulations of this sampling space and we give informal definitions of them in the following paragraphs.

Input space. The indirect manner in which to sample a space of the vehicle states is to sample the input space of the system. The sampled input would then induce the vehicle states according to the models in Sections 2.1 and 2.2. The actual input space being sampled depends on the model being used: when using the state space representation of Section 2.1, the input space is the *state input space*; when using the maneuver automaton representation of Section 2.2, the input space is the *maneuver space*.

The *state input space* is the product of the input space of the state space system in Section 2.1, $\mathcal{U}$, and the prediction period, $\mathcal{T} = [t_0, t_f]$, or $\mathcal{U} \times \mathcal{T}$. The state input space is usually sampled by discretising the state space system of Section 2.1 for a fixed sampling period ΔT. The input to the discretised state space system is held constant over each sampling period and we choose a value from the input space $\mathcal{U}$ for each sampling period. The dimension of the state input space is usually quite large, which causes the sampling of the state input space to be complex.

The *maneuver space*, $\mathcal{M}$, is the input space of the maneuver automaton representation which was introduced by Frazzoli [23–26] and which is discussed in Section 2.2. The input to a maneuver automaton is the ordered pair consisting of the coasting time of the current trim trajectory and the next maneuver to jump to, which is a member of the set Q_M. The maneuver space is therefore given by $\mathcal{M} = \mathbb{R}^+ \times Q_M$. The dimension of the maneuver space is much smaller than that of the state input space, which simplifies the sampling of the input space. This reduced complexity comes at the cost of reduced dynamic capability as discussed in Section 2.2

State space. Another manner in which to sample a space of the vehicle states is directly sampling the vehicle state space or a subset thereof. When applying this approach, we generate samples in the state space and attempt to connect them by constructing inputs to the system that would steer the vehicle from one state to the other. For a system with differential constraints, this is a much more complex problem than calculating the states induced by a given input. The connection of states by means of feasible paths is the task of the local planning method (Subsection 6.2.4).

One formulation of the sampling space is the the continuous vehicle *state space*, $\mathcal{X}$, as defined in Section 2.1. Using the full state space ensures that all vehicle states can be sampled, subject to the constraints, but due the typically large dimensions of the state space, sampling the full state space is usually intractable.

A robot or vehicle *configuration* is the collection of states necessary to describe the robot or vehicle position and orientation from an inertial reference system and is a subset of the robot or vehicle states. The set of all possible configurations is called the *configuration space* $\mathcal{C} \subset \mathcal{X}$ [48, 71] and is a popular sampling space used in robotics. This is the smallest subset of the state space which is also used as a sampling space and the sampling problem is relatively simple, but it might be an unsuitable choice when the vehicle dynamics are considered.

When the dynamics of a vehicle are important to be considered for motion planning, the *phase space* [48], which is the set of all the possible vehicle configurations and their derivatives, is often employed. The phase space is also a subset of the state space.

[2] A *feasible path* is a vehicle path that satisfies both differential and global constraints.

When the environment consists of moving obstacles, it is not sufficient to only sample the chosen sampling space, because a reachable vehicle state at one time instant might not be reachable at another time instant. It is therefore necessary to sample time as well as vehicle states. When the state space is used for sampling, the full sampling space is then given by $\mathcal{X} \times \mathcal{T}$.

6.2.2 Deterministic and Random Sampling

One manner in which to approach the problem of sampling the chosen space is to construct a grid of points across the sampling space. If there is a way of connecting the points on the grid with feasible paths, then classical AI search methods like dynamic programming, A* search or bidirectional search [71] can be used to find a feasible path from the initial to the goal states. However, for high dimensional sampling spaces, the number of grid points with sufficiently high resolution becomes excessive, which causes the problem to become intractable. To overcome this so-called curse of dimensionality, researchers have resorted to generating random samples and then connecting them in various ways. Most existing kinodynamic planners use random sampling, such as the Probabilistic Roadmap methods (PRMs) [25, 39], Rapidly-Exploring Random Trees (RRTs) [50], the method of Hsu et al. [33] and the Randomized Potential Field Planner (RPP) [7].

Recently, researchers have investigated using deterministic sampling that performs as well as or better than random sampling [29, 49, 51]. The advantage of random sampling is its simplicity, whereas deterministic sampling usually provides a more uniform sample distribution.

6.2.3 Path Cost

When confronted with a choice of two vehicle trajectories to the same state, we want to have some way to choose the best trajectory. For this purpose, we associate a cost function J with each trajectory, assumed to be in the form[3]

$$J(\mathbf{y}, \boldsymbol{\mu}) \triangleq \int_{t_0}^{t_f} \gamma(\mathbf{y}(\tau), \boldsymbol{\mu}(\tau)) \, \mathrm{d}\tau \tag{6.1}$$

with $\mathbf{y}(t_0) = \mathbf{y}_0$ and the interval $[t_0, t_f]$. It is possible to construct an incremental cost function $\gamma(\cdot)$ for a wide variety of optimal control problems, such as minimum-time, minimum-energy, minimum-path-length and minimum-fuel problems [26].

Kochenderfer et al. [41] includes the probability of conflict in the definition of the cost function and consider it a parameter to be minimised. We rather view the probability of conflict as a constraint, meaning that the probability of conflict for a certain path should fall below a threshold in order for the path to be considered safe. This corresponds to a non-negotiable safety requirement.

6.2.4 Local Planning Methods and Metrics

Many sampling-based planning methods rely on a method to connect two state-time pairs – say $(\mathbf{y}_1, t_1)$ and $(\mathbf{y}_2, t_2)$ – optimally with respect to the cost function J and subject to differential constraints (global constraints are neglected in this case) by designing the input $\boldsymbol{\mu}(t)$ for $t \in [t_1, t_2]$. This method is called a *local planning method* (LPM) and the problem a *two-point boundary value problem* (BVP) [48]. For some systems, this is difficult to solve and may be computationally expensive. When the chosen sampling space is the maneuver space, the output of the LPM is a sequence of motion primitives that is optimal in the maneuver space, but not

[3] In Part II, we are only interested in the nominal vehicle states of the system in Equation 2.14. The nominal vehicle states are given by $\mathbf{y}(t)$ and the nominal continuous states – which are a subset of the vehicle states $\mathbf{y}(t)$ – are given by $\mathbf{x}(t)$.

necessarily optimal in the full state space. Due to the potentially large computational cost in solving the BVP, some sampling methods are designed to work without a LPM or to call the LPM as little as possible.

The optimal cost-to-go between two state-time pairs is the ideal manner in which to define distance in the sampling space [48]. The only way to accurately calculate this metric for most systems is to solve the BVP. When the BVP is expensive to solve, it is often worthwhile to calculate an approximation to this metric. It is often possible to efficiently calculate a lower bound of the metric, which can be used in A*-type searches, when some constraints are neglected.

6.2.5 Admissibility, Feasibility and Reachability

An *admissible* input is an input function $\boldsymbol{\mu} : [t_0, t_f] \to \mathcal{V}$ in which all the differential constraints are satisfied. We define the set of all admissible inputs as

$$\mathcal{A}_{\boldsymbol{\mu}} \triangleq \{\boldsymbol{\mu}(t) : \forall t \in [t_0, t_f], G\left(\varphi_{\boldsymbol{\mu}}(\mathbf{y}(t_0), t), \boldsymbol{\mu}(t)\right) \leq \mathbf{0}\}. \tag{6.2}$$

A *feasible* input is an admissible input for which the induced vehicle trajectory has a probability of conflict below the threshold for safe operation. We define the set of all feasible inputs as

$$\mathcal{F}_{\boldsymbol{\mu}} \triangleq \left\{\boldsymbol{\mu}(t) \in \mathcal{A}_{\boldsymbol{\mu}} : P_{C,\boldsymbol{\mu}}(t_f) < P_C^{\mathrm{MAX}}\right\}, \tag{6.3}$$

where $P_{C,\boldsymbol{\mu}}(t_f)$ is the probability of conflict of the vehicle path induced by the input $\boldsymbol{\mu}(t)$. We also define the set of feasible inputs that would steer the vehicle path to the state-time pair $(\mathbf{y}_1, t_1)$ as

$$\mathcal{F}_{\boldsymbol{\mu}}(\mathbf{y}_1, t_1) \triangleq \{\boldsymbol{\mu}(t) \in \mathcal{F}_{\boldsymbol{\mu}} : \varphi_{\boldsymbol{\mu}}(\mathbf{y}(t_0), t_1) = \mathbf{y}_1\} \tag{6.4}$$

and the set of feasible inputs that would steer the vehicle path to an element in the set $\mathcal{Y}_1$ as

$$\mathcal{F}_{\boldsymbol{\mu}}(\mathcal{Y}_1) \triangleq \{\boldsymbol{\mu}(t) \in \mathcal{F}_{\boldsymbol{\mu}} : \exists (\boldsymbol{\xi}, \tau) \in \mathcal{Y}_1, \varphi_{\boldsymbol{\mu}}(\mathbf{y}(t_0), \tau) = \boldsymbol{\xi}\} \tag{6.5}$$

A state-time pair $(\mathbf{y}_1, t_1)$ is said to be *reachable* from $(\mathbf{y}_0, t_0)$ if there exists a feasible input that steers the vehicle from $(\mathbf{y}_0, t_0)$ to $(\mathbf{y}_1, t_1)$. This definition is similar to that of Frazzoli et al.[25] and Hsu et al.[33]. We define the *reachable set* $\mathcal{R}(\mathbf{y}_0, t_0)$ as all the state-time pairs that are reachable from $(\mathbf{y}_0, t_0)$, or

$$\mathcal{R}(\mathbf{y}_0, t_0) \triangleq \{(\boldsymbol{\xi}, \tau) : \exists \boldsymbol{\mu}(t) \in \mathcal{F}_{\boldsymbol{\mu}}(\boldsymbol{\xi}, \tau), \varphi_{\boldsymbol{\mu}}(\mathbf{y}(t_0), t_0) = \mathbf{y}_0\} \tag{6.6}$$

Another useful definition is that of the *time-indexed reachable set*, $\mathcal{R}_\tau(\mathbf{y}_0, t_0)$, which is the set of all reachable states at time τ, or

$$\mathcal{R}_\tau(\mathbf{y}_0, t_0) \triangleq \{(\boldsymbol{\xi}, \varsigma) \in \mathcal{R}(\mathbf{y}_0, t_0) : \varsigma = \tau\} \tag{6.7}$$

6.2.6 Probabilistic Completeness and Convergence

As mentioned in Subsection 6.1.2, sampling-based planning methods are not complete. However, providing a weaker guarantee is often possible, namely *probabilistic completeness*. A sampling-based method is probabilistically complete if the probability of finding a feasible path from the initial to the goal states, if such a path exists, tends to one as the number of samples tends to infinity.

In addition to asserting probabilistic completeness, Hsu et al. [33] proved the probability of their method not finding a feasible path, if one exists, decreases exponentially in the number of samples. They introduced the concept of *expansiveness* in state-time space which is a measure of the difficulty of capturing the connectivity of the state-time space by random sampling. It is extremely problematic though to characterise an environment in terms of expansiveness.

6.3 Kinodynamic Planning Problem Definition

With the important concepts of Section 6.2 in hand, we now give a definition of the kinodynamic planning problem before reviewing existing sampling-based algorithms in Section 6.4.

Define the optimal feasible inputs as the set of feasible inputs for which the cost function J is minimised, or

$$\mathcal{F}^*_{\boldsymbol{\mu}}(\mathcal{Y}_F) \triangleq \{\boldsymbol{\mu}(t) \in \mathcal{F}_{\boldsymbol{\mu}}(\mathcal{Y}_F) : \forall \boldsymbol{\psi}(t) \in \mathcal{F}_{\boldsymbol{\mu}}, J(\varphi_{\boldsymbol{\mu}}, \boldsymbol{\mu}) \leq J(\varphi_{\boldsymbol{\psi}}, \boldsymbol{\psi})\} \tag{6.8}$$

The kinodynamic motion planning problem is then finding a vehicle input in the optimal feasible vehicle input set: $\boldsymbol{\mu}^*(t) \in \mathcal{F}^*_{\boldsymbol{\mu}}(\mathcal{Y}_F)$.

As discussed in Subsection 6.1.2, finding the optimal feasible input is an intractable problem for a system with differential constraints. We therefore settle for an approximation of the optimal input by using sampling techniques.

A watered-down version of the kinodynamic planning problem is to find any feasible input $\boldsymbol{\mu}(t) \in \mathcal{F}_{\boldsymbol{\mu}}(\mathcal{Y}_F)$ – we will not use this definition, but rather attempt to find an approximate optimal input.

6.4 Overview of Existing Methods

Having established a formal problem definition (Section 6.3) and having discussed important motion planning concepts (Section 6.2), we proceed to give a review of existing sampling-based kinodynamic planning methods.

Existing kinodynamic planning methods can broadly be divided into *incremental search methods* and *roadmap methods*. We give an overview of the important methods below.

6.4.1 Incremental Search Methods

The incremental search methods sample the vehicle state space indirectly by sampling the input space. These methods incrementally expand a set of known reachable points by applying a simple input command, which can be a single motion primitive or a state space input held constant over a short time period, from a selected known reachable point. If the induced vehicle path is feasible, the vehicle state at the end of this path is added to the known reachable set. This process is repeated until a reachable point close to the goal state is generated or a time limit is reached. These methods make little or no use of a LPM and are therefore attractive when solving the BVP is computationally expensive.

One of the earlier incremental search methods is the Randomised Path Planner (RPP) of Barraquand and Latombe [6, 7]. The RPP discretises the workspace and configuration space and constructs a numerical potential field in the configuration space. It then constructs a tree of feasible paths by searching towards the lowest potential in the configuration space and uses a random walk to escape local minima. The initial formulation [7] only allowed holonomic planning, but subsequent work [8] extended the algorithm to nonholonomic robots. The construction of the numerical potential field relies on an explicit obstacle description and makes this method unsuitable for uncertain environments.

A more recent incremental search method for kinodynamic planning is that of Hsu et al. [33]. This method generates new reachable points by choosing a known reachable point to extend, choosing a random input signal and then simulating the system for a short time period. The newly generated path segment is then tested for conflict and if no conflict is predicted, the point at the end of the path section is added to the set of known reachable points. This process is repeated until a reachable point within an endgame region is generated. The choice of which known reachable point to extend at each step is made by weighted random sampling where the weight of each known reachable point is inversely proportional to the density of points

in its vicinity. This weighted sampling prevents oversampling in any region of the sampling space. The sampling space is the state-time space and this method is therefore suitable for dynamic environments. This method assumes no uncertainty in the current environment states and handles uncertainty in the future obstacle trajectories by replanning at fixed intervals. The method is proven probabilistically complete for an idealised input sampling function where a uniformly distributed random input will produce uniformly distributed vehicle states. In addition, the probability of not finding a feasible path, if one exists, is proven to converge exponentially to zero in the number of known reachable points. In practice, this ideal input sampling function is difficult to construct. The method explores the reachable set of points from the initial states without favouring any specific region of the sampling space – in environments with wide open spaces, this method might therefore appear aimless. An advantage of this method is that it does not make use of a LPM at all.

6.4.2 Roadmap Methods

The roadmap methods expand the set of known reachable points by sampling the vehicle state space directly. They choose points in the vehicle state space – also called *milestones* – and then attempt to connect these milestones to known reachable points using feasible paths. These methods employ a LPM to connect the milestones. Because the roadmap methods heavily relies on the LPM, they are suited to those problems where the BVP can be solved efficiently.

Most roadmap methods use a random function to generate milestones. This class of roadmap methods is called *probabilistic roadmap methods* (PRMs) and was introduced for holonomic systems by Kavraki et al. [39] with similar earlier work done by Overmars [54]. The PRM was originally designed for multiple queries in static configuration spaces. In its original formulation, the method consists of two phases: the preprocessing phase and the query phase. In the preprocessing phase, a number of milestones in the free configuration space are randomly generated, connected together using collision-free paths and stored in a unidirectional graph – this is executed offline beforehand. In the query phase, the initial and goal configurations are connected to the graph and a feasible path between the initial and goal configurations is found by searching the graph – this is executed online. If the configuration space is static, multiple queries can be run for a single preprocessing phase.

The PRM has been extended to kinodynamic planning in dynamic environments. These extended PRMs differ from the original PRM in the following ways: firstly, the sampling space is the full state-time space – or some subspace thereof – instead of only the configuration space; secondly, the generation of milestones *and* their connection to the initial and goal points are done online instead of using an offline preprocessing phase and an online query phase; thirdly, only a single query is done for each batch of milestones instead of doing multiple queries per batch; fourthly, milestones are connected together using a LPM instead of using simple geometric methods, and fifthly, *lazy conflict checking* is implemented where only the path being queried is tested for conflict instead of testing all the interconnecting paths before any attempt at finding a feasible path to the goal is made. We discuss two important kinodynamic PRM methods below.

One kinodynamic PRM method is the Rapidly-Exploring Random Tree (RRT) [47, 48, 50]. This algorithm randomly generates a milestone at each algorithm iteration from a uniform distribution of the sampling space and selects the *nearest* point out of a set of known reachable points with respect to some distance metric (which is often the Euclidean distance). It then chooses a constant input signal that generates a path segment from the nearest known reachable point that brings the system as close as possible to the chosen milestone. This path segment is then tested for conflict – if no conflict is predicted, the point at the end of the path segment is added to the set of known reachable points. Alternatively, if conflict is predicted, a milestone is picked on the path segment just before the predicted conflict and then added to the set of

known reachable points. The RRT is proven probabilistically complete in the sampling space. It only attempts to find a feasible path and does not optimise the feasible path in any way. The RRT description defines the sampling space as the state space (or subspace thereof) and is therefore only suitable for static environments. However, it is simple to extend the method to dynamic environments by defining the sampling space as the state-time space (or subspace thereof).

Another kinodynamic PRM method was introduced by Frazzoli [23–25], based on the RRT method, but extending it in a number of ways. Similar to the RRT method, this method randomly generates a milestone from a uniform distribution of the sampling space at each algorithm iteration and then attempts to connect this milestone to a set of known reachable points. Differently from the RRT method, it attempts to connect the milestone *exactly* to a known reachable point and attempts to connect the milestone to *each* point in the set of known reachable points until it finds a feasible path to the milestone. The method sorts the sequence in which the known reachable points are tested for connectivity to the randomly generated milestone in ascending order according to the cost-to-go between each point and the milestone in terms of the cost function J. If the method finds a feasible path between a known reachable point and the milestone, it adds the milestone to the set of known reachable points as well as a secondary milestone chosen at a random point along the path segment. It then attempts to connect this new milestone to the goal state using a LPM. When a feasible path to the goal is found, the method prunes the tree of reachable points that would never yield a lower total cost to goal than the found path. The method continues to generate milestones in order to find feasible paths with lower total costs to goal until a time limit is reached – it therefore searches for an optimal path.

An often-cited roadmap method that does not use randomisation is the Ariadne's Clew algorithm of Mazer et al. [53]. This method generates a milestone at each iteration and then tries to connect the milestone to the goal state. The milestones are uniformly distributed by choosing the new milestone such that the shortest path length between the existing milestones and the new milestone is maximised. The path length between milestones is determined by using a LPM that bounces off obstacles. The Ariadne's Clew algorithm is only applicable to holonomic systems and it is generally very difficult to place a new milestone [48].

Having supplied an overview of the existing kinodynamic planning methods, we proceed in Chapter 7 to present a method that extends the existing methods and which performs well in uncertain, dynamic and cluttered environments.

Chapter 7

A Motion Planner for Conflict Resolution

Chapter 6 discusses the approaches to conflict resolution and provides an overview of existing sampling-based kinodynamic planning methods. From this overview, it is evident that most methods are designed for cluttered environments, but few can handle dynamic environments and none are able to manage uncertain environments in an integrated and robust way. In addition, there are few guidelines available for choosing the size of the sampling region, which could impact severely on the method performance if chosen unwisely. In this chapter, we propose some changes to the existing sampling-based motion planning methods to enable them to manage cluttered, dynamic and uncertain environments. We also provide a guideline to choose and adapt the sampling region.

This chapter leads off by extending PRM methods to deal with uncertain, cluttered and dynamic environments (Section 7.1) after which we propose a method to choose the size of the sampling region (Section 7.2). Lastly, we combine the first two sections and present a motion planning algorithm in Section 7.3.

7.1 Kinodynamic Planning in Uncertain, Cluttered and Dynamic Environments

In this section, we discuss some problems that existing motion planning methods experience in uncertain, cluttered and dynamic environments and propose some changes to improve their performance.

From the overview of existing methods in Section 6.4, we identify three candidate methods for conflict resolution: the incremental search method of Hsu et al.[33] and two probabilistic roadmap methods – the rapidly-exploring random tree (RRT) method[47] and the method of Frazzoli et al.[25].

At first glance, the method of Hsu et al.[33] seems attractive because it does not solve the BVP and therefore does not have a LPM, which could be a computationally expensive procedure. In addition, when the maneuver automaton representation of Section 2.2 is used, the sampling space is low-dimensional. The known reachable points are also indexed by time, making the method suitable for dynamic environments. However, from an example implementation, we observed several disadvantages, which are detailed below.

1. The method appears undirected – there is no bias towards the goal states due to the formulation that ensures probabilistic completeness. When the state space contains large regions of free space, the method is therefore quite slow to find a feasible path to a point near the goal state.

2. The method requires a sampling procedure to sample the input space such that the induced vehicle states are uniformly distributed. It is hard to design such a procedure, especially when the maneuver automation representation is used.

3. It is difficult to calculate the point density near a specific point. This density is used to weight the known reachable points which is used in their selection for expansion. Hsu et al.[33] calculated the density by counting the number of points located less than a fixed distance from the specific point. As the number of known reachable points increase, the process of updating the density becomes computationally more expensive. Choosing the fixed distance used in the density calculation is also problematic.

These disadvantages and the fact that the method performed worse than PRM methods in a comparative simulation for a dynamic environment [25] make PRM methods more attractive for conflict resolution.

The PRM methods under consideration do not manage all uncertain, cluttered and dynamic environments satisfactorily in their original formulation. We discuss the problems these methods encounter and propose some changes for each type of environment in the next three subsections.

7.1.1 Uncertain Environments

The main contribution of this chapter comprises extending the existing kinodynamic motion planning methods to work in uncertain environments. Before avenues to achieve this are explored, we present an overview of previous attempts to handle uncertain environments.

The only kinodynamic planning methods that we found which address uncertainty in the environment is that of Hsu et al. [33] and Fulgenzi et al. [27]. The method of Hsu et al. [33] attempts to treat uncertainty in the future obstacle positions in two ways: firstly, it assumes that the obstacles move in straight lines, but increases the obstacle boundaries as the prediction time increases; and secondly, it repeatedly replans at fixed intervals. The former corresponds to the worst-case conflict detection approach and makes this method overly cautious; the latter results in the autonomous vehicle avoiding most conflict that would be caused by obstacles changing direction, but it is a rather ad hoc approach and fails to provide any guarantee of a feasible path. The method of Fulgenzi et al. [27] constructs a probabilistic occupancy grid for static obstacles and a list of Gaussian processes for moving obstacles. It then explores the vehicle configuration space using an extension to the Rapidly-exploring Random Tree (RRT) algorithm. It tests each candidate path segment for collision by calculating the maximum instantaneous probability of collision along this segment. The total probability of collision for a certain path is then the combination of the probability of collision of its segments. The maximum instantaneous probability of collision is a lower bound to the actual probability of collision as discussed in Subsection 3.2.1, therefore this method might produce somewhat unsafe paths in certain cases.

We propose a manner to handle uncertainty that is both simple and powerful. This is made possible by the efficient probabilistic conflict detection method developed in Part I. In this approach, we replace the conflict detection module for a system without uncertainty with a probabilistic conflict detection module. Because sampling-based planning methods are separated from the environment description by the conflict detection module, extending the conflict detection module to address uncertain environments effectively extends the planning method to manage uncertain environments. The diagram in Figure 7.1, which is based on a figure in the book by LaValle [48], illustrates the separation of the sampling-based planning method from the environment model for the standard case where there is no uncertainty in the vehicle or environment model. The diagram in Figure 7.2 illustrates the way in which we extend the sampling-based planning methods to address uncertainty in the environment and vehicle models by using a probabilistic conflict detection module. When the differences between Figures

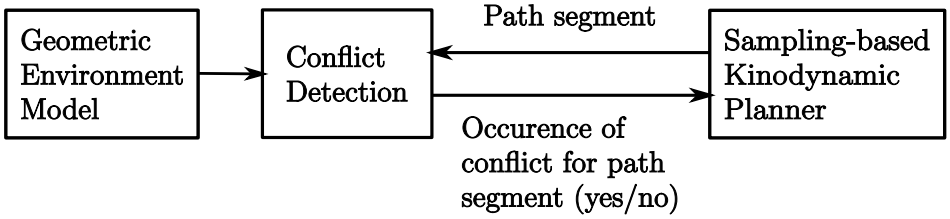


Figure 7.1: Separation of sampling-based planners from environment model by conflict detection module (case without uncertainty in the environment and vehicle models)

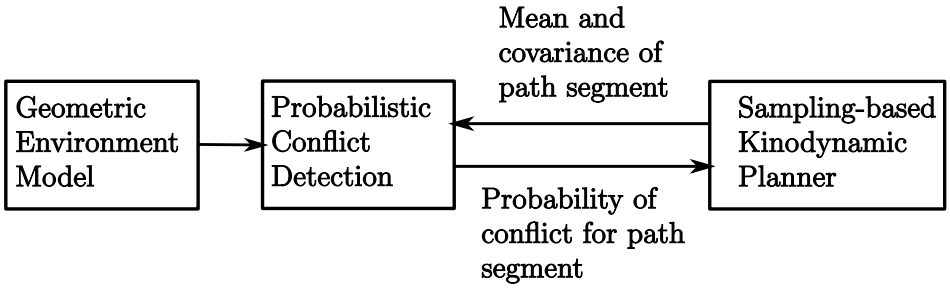


Figure 7.2: Separation of sampling-based planners from environment model by probabilistic conflict detection module (case with uncertainty in the environment and vehicle models)

7.1 and 7.2 are studied, it becomes clear that we can employ any sampling-based kinodynamic planner for an uncertain environment and vehicle model as long as the only interface from the planner to the environment model is at the conflict detection module and the following three issues are resolved:

1. A conflict detection test takes much longer to compute for an environment and vehicle model with uncertainty than for a model without uncertainty.

2. The planner has to generate the mean and covariance of the path segment it wants to test for conflict.

3. The probabilistic conflict detection module returns a probability of conflict compared to a binary pass/fail answer of the conflict detection module without uncertainty.

Sánchez and Latombe [72] reported conflict tests between two bodies without uncertainty that are described by triangulated surfaces with 500 000 triangles executing in less than 2 milliseconds (with preprocessing) while the three-dimensional example with uncertainty in Section 5.2 with 258 triangles executed in almost 100 milliseconds (without preprocessing). A planning algorithm typically spends most of its execution time in the conflict detection module. With model uncertainty, the conflict checker can evaluate significantly fewer path sections in a given time interval than without model uncertainty. It then becomes worthwhile to spend more processing time to ensure the generation of good quality candidate milestone and path segments to test for conflict.

The requirement for the planner to generate the mean and covariance of the vehicle states for the path segment to be tested is usually unproblematic to comply with. Subsection 2.1.2 details the calculations for time-variant linear systems.

The fact that the probabilistic conflict detection module returns a probability of conflict for the tested path segment adds a new parameter to each milestone in the set of known reachable points. A feasible path from the initial to goal states requires the probability of conflict for the *whole* path to be below a certain threshold. It is therefore necessary to store the accumulated conflict probability from the initial state at each milestone. When we have to choose between

different paths to the same milestone, the choice is now not only dependent on the accumulated cost function value, but also on the accumulated conflict probability.

The proposed manner to address uncertainty is consistent with the architecture of the conflict avoidance system presented in Section 1.3. The interaction between the conflict detection module and the conflict resolution module contains no dissonance because the motion planner uses the probabilistic conflict detection method when designing a feasible path. All feasible paths produced by the conflict resolution module therefore comply with the maximum conflict probability constraint.

7.1.2 Dynamic Environments

When the environment contains moving obstacles, only sampling the state space or subspace thereof do not ensure probabilistic completeness – to retain probabilistic completeness, we have to sample the state-time space. The idea of sampling the configuation-time space to handle multiple moving robots was introduced by Erdmann and Lozano-Pérez [18]. This idea was extended to sampling the state-time space by Fraichard [21]. Of the candidate kinodynamic planners, the RRT method only samples a reduced state space [50], the method of Hsu et al. [33] indirectly samples the full state-time space and method of Frazzoli [24] only samples a reduced state space.

Adding the time dimension to the sampling space increases the complexity of the problem. However, it is evident that vehicle paths may only move in one direction in the time dimension. We exploit this feature to reduce the complexity of the rejection sampling procedure detailed in Section 7.2.

The choice of which states to include in the subspace of vehicle states in the sampling space is also complicated. When generating a sample in the reduced state-time space, we want its reachable set to stay fairly constant regardless of the value of the omitted states at the sample point. On one hand, it is preferable to include many states in the sampling space to satisfy the preceding requirement; on the other hand, the number of states in the sampling space has to be kept to a minimum to reduce the problem complexity. Frazzoli [24] defined a sampling space from a symmetry defined on the system. We prefer to leave the definition of the sampling space as a design choice that depends on the specific environment and vehicle dynamics.

7.1.3 Cluttered Environments

The choice of the number of samples to generate for PRM planners is determined by the trade-off between the difficulty of capturing the connectivity of the sampling space and the computational complexity: generating more samples increases the chance of finding a feasible path, but also increases the execution time of the algorithm. The computational cost of testing the reachability of a milestone increases as the size of the set of known reachable points increase due to operations such as finding the nearest known reachable point (RRTs [50]), sorting the list of known reachable points (Frazzoli et al. [24]) or attempting to connect the milestone to every known reachable point until a feasible path to the candidate milestone is found (Frazzoli et al. [24]). This feature is another motivation for choosing milestones wisely.

One feature of existing kinodynamic PRM planners is that they generate a single milestone for each algorithm iteration and attempt to connect it to the set of known reachable points directly thereafter [24, 50]. This concept is illustrated for 10 milestones in Figure 7.3. The milestones are numbered according to the order in which they are generated. When the conflict probability of a path segment is higher than the chosen threshold, a secondary milestone is created at a point along the path segment such that the accumulated conflict probability at this secondary milestone is below the threshold. The secondary milestone is then added to the set of known reachable points. This secondary milestone creation is also a common feature in existing kinodynamic PRM planners [24, 50]. Two effects of these features include clustering of the

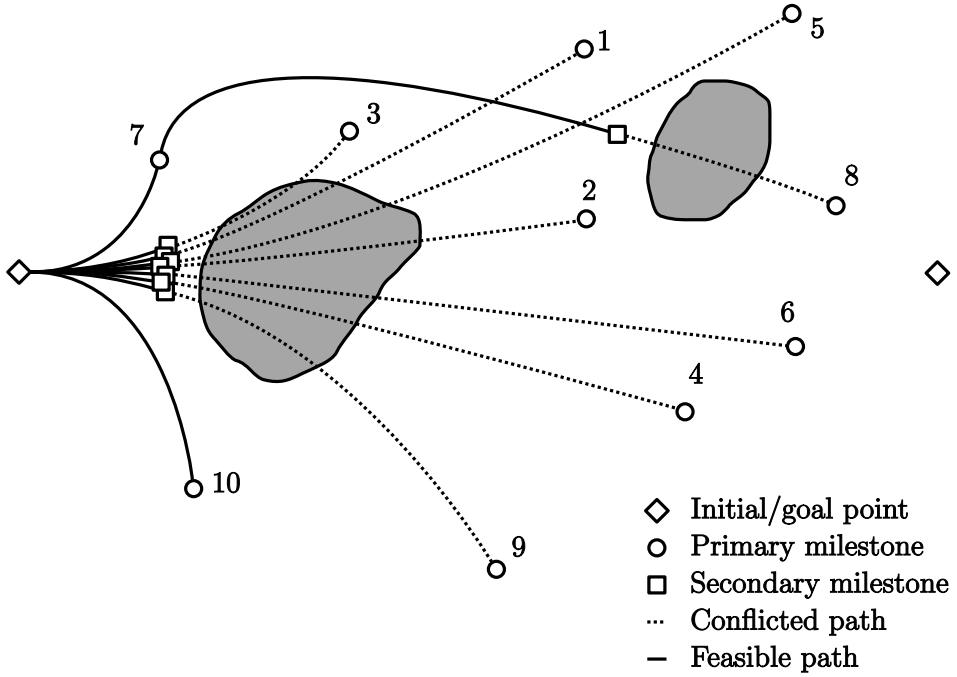


Figure 7.3: Conceptual illustration of single point generation PRM

known reachable points around obstacles and badly-captured connectivity of the sampling space, especially during the initial algorithm iterations. The first effect is undesirable because PRM methods usually strive for a uniform milestone distribution (except when specialised sampling techniques are used to improve connectivity in regions where the connectivity is difficult to capture [10, 32, 85]). The second effect is also clearly undesirable, but especially so when conflict tests are computationally expensive and the number of milestones that can be generated, connected and tested are limited by the allowed execution time.

A possible solution to the abovementioned problems is to adapt the original two-staged PRM formulation for online kinodynamic planning. According to this approach, the algorithm generates a batch of milestones at each program iteration, sorts them according to some metric (ideally cost-to-go from initial state) and then attempts to connect them to the set of known reachable points using the sorted order. This approach is illustrated conceptually in Figure 7.4. The milestones are numbered according to their sorted order. We also do away with the secondary milestone creation. When we compare Figure 7.4 with Figure 7.3, it is evident that the connectivity of the sampling space is captured is much improved. This effect is due to sorting the batch of milestones: the chance of being able to connect a milestone increases as the distance of the shortest path between the milestone and the set of known reachable points decreases. Attempting to first connect the milestones that are closer (in terms of cost-to-go) to the initial state has the effect of keeping the path segments being tested for conflict short. This increases the chance that the milestone will be connected, which in turn increases the chance that other milestones will be connected. Omitting secondary milestones ensures that the set of reachable points only contains milestones that are sampled from a uniform distribution, ensuring better placement of the known reachable points.

7.2 Bounding the Sampling Region

One aspect of PRM methods that is usually not dealt with when new methods are proposed is how to bound the sampling region. The bounds of the sampling region are important: making a choice of too large bounds may cause the algorithm to unnecessarily explore remote sections of

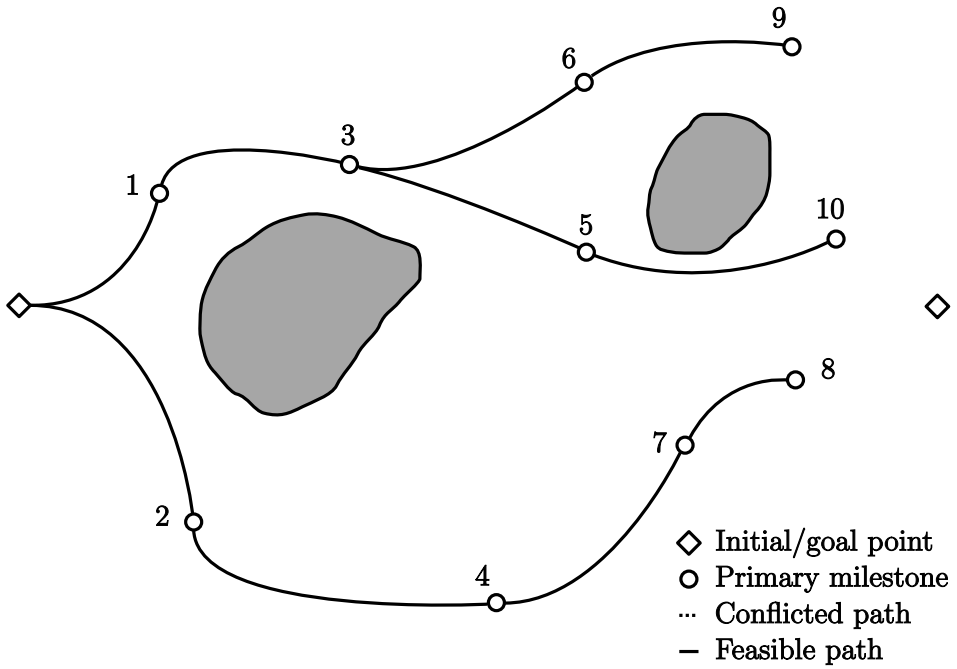


Figure 7.4: Conceptual illustration of batch point generation PRM

the free space, whereas choosing the bounds too small may prevent the algorithm from finding a feasible path.

A recently proposed method that adjusts the sampling region is the dynamic-domain RRT [35, 86]. This method uses rejection sampling to retain only those samples that lie within a certain radius of a known reachable point. This negates the effect that different sampling region sizes have on the method and dramatically improves the algorithm performance in certain cases.

We propose another way to bound the sampling region in such a way that the optimal path cost to connect each point in the sampling region to the initial and goal states are bound by a maximum cost value. This maximum cost value is then adapted at various stages of the algorithm execution. Because this approach bounds the optimal path cost between the initial and goal states via the sampled point, it can be used to bias the sampled points towards the goal state. This directed character can also be effected using best-first search – such as A* search – but at the loss of probabilistic completeness, whereas bounding the sampling region retains probabilistic completeness. We firstly present the ideal way to bound the sampling region after which we discuss some techniques to increase the sampling procedure efficiency.

Let $\boldsymbol{\mu}^{\text{opt}}_{(\mathbf{y}_1,t_1):(\mathbf{y}_2,t_2)}(t)$ be the admissible input that is optimal with respect to the cost function J and connects the state-time pairs $(\mathbf{y}_1, t_1)$ and $(\mathbf{y}_2, t_2)$. Similarly, for the set of state-time pairs, $\mathcal{Y}_2$, let $\boldsymbol{\mu}^{\text{opt}}_{(\mathbf{y}_1,t_1):\mathcal{Y}_2}(t)$ be the optimal input that corresponds to $\boldsymbol{\mu}^{\text{opt}}_{(\mathbf{y}_1,t_1):(\boldsymbol{\xi},\tau)}(t)$ for some $(\boldsymbol{\xi}, \tau) \in \mathcal{Y}_2$ such that the cost function J is minimised over $\mathcal{Y}_2$. We now place a constraint on the sampling region such that for each sample $(\boldsymbol{\xi}, \tau)$, the sum of the cost function value for the optimal input from the initial state $\boldsymbol{\mu}^{\text{opt}}_{(\mathbf{y}_0,t_0):(\boldsymbol{\xi},\tau)}(t)$ for $t \in [t_0, \tau]$ and the optimal cost-to-go – corresponding to the input $\boldsymbol{\mu}^{\text{opt}}_{(\boldsymbol{\xi},\tau):\mathcal{Y}_F}(t)$ for $t \in [\tau, t_f]$ – is less or equal than a specified value J_B. Informally, this constraint ensures that for *each* point in the sampling space, the lowest cost of a path connected to the sampled point and which satisfies the differential constraints will be at the most J_B. In addition, because this is the only constraint on the sampling space, the sampling space will contain *all* the points for which the least connection costs are at the most J_B. Since we employ cost as the measure to rank the paths, we concern ourselves therefore only with points that might provide us with a better path than a certain value.

Implied in the constraint above is another constraint induced by the structure of the sam-

pling space: because of the time dimension in the state-time space and the differential constraints on the system, not every state-time pair can be connected to the initial and goal states. Samples should therefore only be chosen in regions where a point can *at least* be connected to both the initial state and one of the final states.

Usually, the only manner in which to calculate the least connection cost is to solve the BVP using a LPM. The simplest way of implement the bounding of the sampling region is to sample uniformly over a region encompassing the bounded region and using rejection sampling to retain those points satisfying the least connection cost constraint, which is calculated using a LPM. The induced constraint due to the inclusion of the time dimension in the sampling space is also satisfied because the LPM tests whether the sampled point can be connected to the initial or goal states.

Because solving the BVP is often computationally expensive, using the above-mentioned rejection sampling strategy might be quite inefficient – it would be more efficient to use a simpler test to accept or reject a sample. We introduce such a simple test by defining a subset of the set of admissible inputs as defined in Equation 6.2. Suppose the differential constraint function $G(\cdot)$ contains N elements (similar to Equation 2.3), then the set of admissible inputs can be written as

$$\begin{aligned}\mathcal{A}_{\boldsymbol{\mu}} &= \left\{ \boldsymbol{\mu}(t) : \forall t \in [t_0, t_f], \bigvee_{i=1}^{N} G_i \left(\varphi_{\boldsymbol{\mu}}(\mathbf{y}(t_0), t), \boldsymbol{\mu}(t)\right) \leq 0 \right\} \\ &\subseteq \left\{ \boldsymbol{\mu}(t) : \forall t \in [t_0, t_f], G_k \left(\varphi_{\boldsymbol{\mu}}(\mathbf{y}(t_0), t), \boldsymbol{\mu}(t)\right) \leq 0 \right\}\end{aligned} \tag{7.1}$$

for some $k \in [1, N]$. We can therefore create a superset of the sampling space by only applying some of the differential constraints. One common differential constraint is the maximum speed constraint. When applying only the maximum speed constraint on the sampling space, we obtain a simple geometric constraint as illustrated for a two-dimensional configuration space in Figure 7.5.

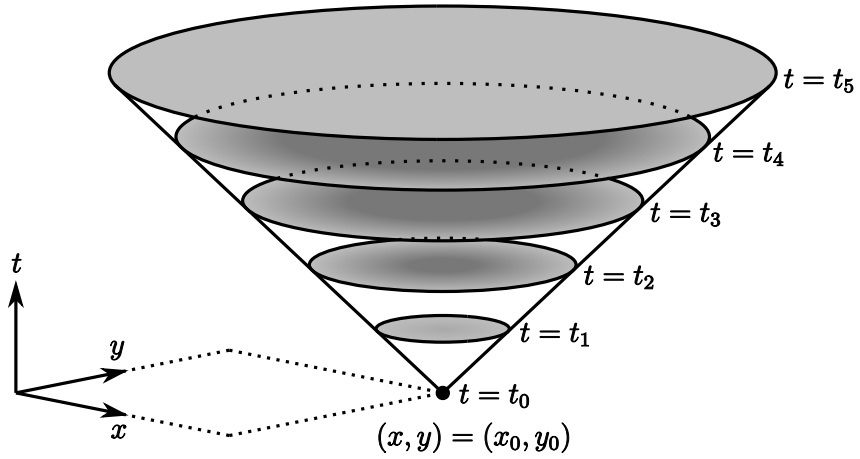


Figure 7.5: Visualisation of maximum speed constraint

The definition of the goal states often includes an explicit limit on the time to goal. When it is not explicitly defined, limiting the value of the cost function will limit the time to goal for common cost function definitions. When a bound for the time to goal is available, we can add another geometric constraint based on the maximum speed constraint, illustrated in Figure 7.6. The diagram shows a conic constraint as presented in Figure 7.5 combined with a constraint that is given by an inverted cone with its origin at the goal state and the time limit.

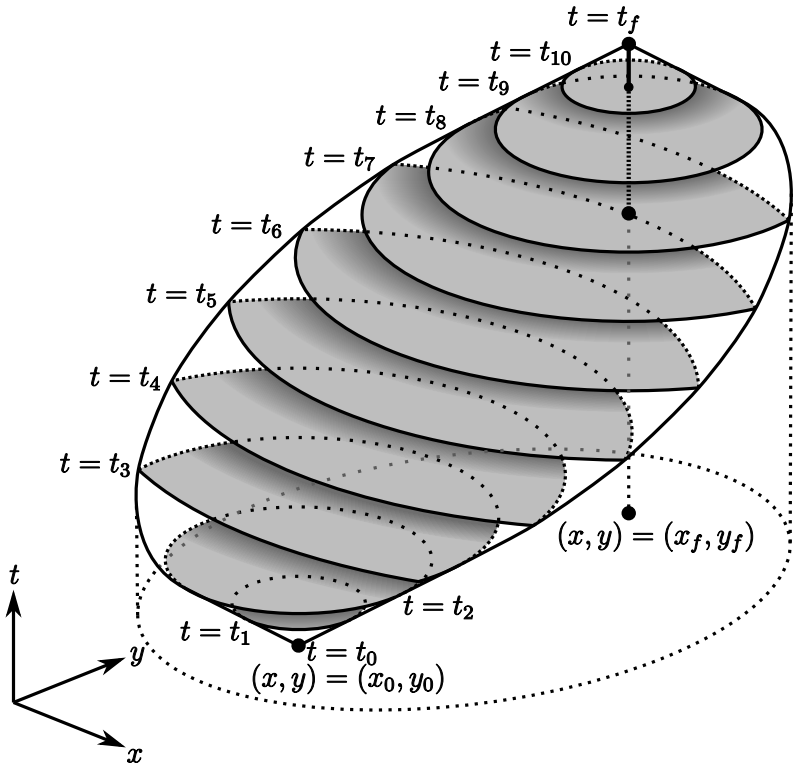


Figure 7.6: Visualisation of maximum speed constraint for bounded time to goal

It is often possible to construct another simple test to increase the efficiency of bounding the sampling region by rejection sampling. This is achieved by defining a lower bound to the cost function J based on the distance of the path connecting the input with the output via the sampled point under consideration. A lower bound on the distance of this shortest connecting path can then be determined by calculating the sum of the distance from the initial state to the sampled point and the distance from the sampled point to the nearest goal state.

In summation, the rejection sampling procedure consists of the following three steps:

1. A sample is randomly generated from a uniform distribution defined on a superspace of the bounded sampling space.

2. The sample is then tested using simple and efficient geometric tests that reject most unsuitable samples.

3. If the sample passes the geometric tests, we create an optimal admissible input connecting the initial and sample points and an optimal admissible input connecting the sample and goal points, using a LPM. If these inputs exist and the combined cost associated with them is lower than J_B, the sample is valid.

We change the value of the least connection cost bound J_B as the algorithm progresses to account for different environments. The rule according to which we adapt the least connection cost bound is detailed in Section 7.3.

7.3 Motion Planning Algorithm

In Sections 7.1 and 7.2, we propose a number of changes to existing PRM methods in order to handle uncertain, cluttered and dynamic environments as well as to negate the detrimental

effect of choosing an unsuitable sampling region. In this section, we combine these changes in a motion planning algorithm.

The proposed motion planner is given in pseudocode in Algorithm 2.

Algorithm 2 MOTION PLANNER

1: $\boldsymbol{\mu}(t) \leftarrow \text{LPM}\left((\mathbf{y}_0, t_0), \mathcal{Y}_F\right)$
2: **if** CONFLICTTEST($\boldsymbol{\mu}(t)$) $< P_C^{\text{MAX}}$ **then**
3: **return** $\boldsymbol{\mu}(t)$
4: **else**
5: Initialise the graph $\mathcal{G}_T$
6: **repeat**
7: Adjust J_B
8: Initialise $\mathcal{Y}_c$ to $\emptyset$
9: **repeat**
10: Sample $(\mathbf{y}_c, t_c)$ from a uniform distribution
11: **if** $(\mathbf{y}_c, t_c)$ in bounded sampling region **then**
12: Add $(\mathbf{y}_c, t_c)$ to $\mathcal{Y}_c$
13: **end if**
14: **until** $\mathcal{Y}_c$ has N_c elements
15: Sort $\mathcal{Y}_c$ according to cost from $(\mathbf{y}_0, t_0)$
16: **for** every element $(\mathbf{y}_c, t_c)$ in $\mathcal{Y}_c$ **do**
17: CONNECT$((\mathbf{y}_c, t_c), \mathcal{G}_T)$
18: **end for**
19: CONNECT$(\mathcal{Y}_F, \mathcal{G}_T)$
20: **if** a new feasible path to goal has been found **then**
21: Prune $\mathcal{G}_T$
22: **end if**
23: **until** allowed execution time has elapsed
24: **return** feasible $\boldsymbol{\mu}(t)$ with lowest associated cost function value
25: **end if**

The different elements of the algorithm are discussed in the following paragraphs.

LPM (line 1). The local planning method (LPM) calculates the optimal admissible input with respect to the cost function J between an initial state-time pair and a goal state-time pair or set of state-time pairs.

CONFLICTTEST (line 2). This is the probabilistic conflict detection test which is the subject of Part I. For this algorithm, we employ a constant conflict probability threshold P_C^{MAX}.

Graph $\mathcal{G}_T$ (line 5). $\mathcal{G}_T$ is the directed graph used to represent the explored portion of the sampling space. The vertices are the known reachable points which are given by state-time pairs and the edges are the feasible inputs connecting the reachable points.

Adjusting J_B (line 7). When no feasible path to goal has been found, the bound on the least connection cost J_B (Section 7.2) is increased by a constant factor at each iteration of the outer loop of the algorithm until a maximum factor of the cost of the optimal admissible path between the initial and goal states as calculated in line 1 is reached. The value of J_B is then held at this maximum level for subsequent iterations. When a feasible path to goal has been found, J_B is set to the cost associated with this path. This ensures that the algorithm does not

generate samples that cannot yield better paths than the one already found. The value of J_B is similarly adjusted when better paths to goal are found.

Rejection sampling (lines 10 and 11). The uniform sampling is done in line 10 using a sampling space suitable for easy sample generation. Those samples that cannot be connected to the initial or goal states or have least connection costs exceeding J_B are then rejected in line 11. We follow the rejection sampling process discussed in Section 7.2 where samples are first tested using simple and efficient geometric tests and the BVP is solved (using the LPM) only for those samples that have passed the geometric tests.

Sorting $\mathcal{Y}_c$ (line 15). The batch of milestones $\mathcal{Y}_c$ are sorted according to the cost function value associated with the optimal admissible path from the initial state. This is the change proposed in Subsection 7.1.3 in order to improve the performance of the algorithm during the initial iterations.

Connect (lines 17 and 19). The Connect function attempts to connect a state-time pair to the graph $\mathcal{G}_T$. We use a similar approach to that of Frazzoli et al.[25]: firstly, the vertices in $\mathcal{G}_T$ are sorted according to some heuristic, and secondly, an attempt is made to connect the milestone to each point in the sorted list of known reachable points until a feasible path to the milestone is found or the list has been exhausted. We use simple geometric tests similar to that found in Section 7.2 to discard the known reachable points whose reachable sets clearly do not contain the milestone. Similar to Frazzoli et al.[25], we apply a sorting heuristic based on the costs of the admissible paths between the known reachable points and the milestone when no feasible path to goal has yet been found. This heuristic ensures that the milestone is connected quickly, enhancing the speed of exploration of the free space. When a feasible path to goal has been found, the focus shifts to optimisation, and we employ a sorting heuristic based on the cost from the initial state – this ensures that the first feasible path found to the milestone is also optimal in the set of known reachable points.

Prune $\mathcal{G}_T$ (line 21). When a feasible path to goal has been found, we do not want to consider those known reachable points that cannot provide a better feasible path. We therefore remove those known feasible points from the graph $\mathcal{G}_T$ for which the sum of the cost from the initial state and the cost of the optimal admissible path to goal exceeds the cost of the feasible path to goal that has already found.

The proposed motion planner generates random samples from a uniform distribution over the bounded sampling space. The planner is therefore probabilistically complete, which means that the probability of finding a feasible path to goal tends to one as the number of generated samples tend to infinity. The bounds on the sampling space change as the maximum least connection cost, J_B, changes throughout the algorithm execution. However, when the planner has not found a feasible path to goal, J_B quickly reaches its maximum value and then stays constant until a feasible path to goal has been found. The probabilistic completeness guarantee is therefore valid for the bound on the sampling space induced by the maximum value of J_B.

Having defined the proposed motion planning algorithm, Chapter 8 proceeds next to illustrate the performance of the method by using the example problems of Chapter 5.

Chapter 8

Conflict Resolution Examples

Chapter 7 proposes various changes to existing kinodynamic motion planning algorithms to extend their application to uncertain, dynamic and cluttered environments. These changes, as well as a proposed bound for the sampling region, are then combined to form a motion planning algorithm. In this chapter, we implement this new motion planning algorithm for the two example problems of Chapter 5.

Section 8.1 illustrates the implementation details and results of the motion planning algorithm applied to the two-dimensional example problem of Section 5.1 and Section 8.2 illustrates the implementation details and results for the motion planning algorithm applied to the three-dimensional example problem of Section 5.2.

8.1 Two Airplanes Example

In this section, we take the two-dimensional example problem of Section 5.1 – that of two airplanes with crossing flight paths – and apply the motion planning algorithm of Chapter 7 for conflict resolution.

8.1.1 Problem Description

For the problem description, we use the same setup as the example in Section 5.1: the planned paths of the airplanes cross at a 90° angle with the uncertainty in their states described by joint Gaussian distributions of which the along-track position uncertainty increases linearly over time.

The conflict resolution problem entails designing an input to the reference airplane such that the probability of conflict between the reference and intruder airplanes falls below the threshold for safe operation. For the construction of an input signal to the reference airplane, we assume the following features of the reference airplane model: the nominal speed of the airplane is constant and the airplane model is described by the maneuver automaton representation of Section 2.2. The maneuver automaton has three trim trajectories: straight flight, left turn with a constant turn radius and right turn with a constant turn radius. The transition times and dynamics between trim trajectories are assumed negligible and the maneuver automaton therefore contains no maneuvers. This model is similar to a version of the Dubins car [15] and a path between two points can be constructed using an efficient geometric procedure.

We choose a minimum-path length cost function which is also, because of the constant nominal airplane speed, a minimum-time cost function.

8.1.2 Implementation

We implemented the motion planner of Chapter 7 as described in pseudocode in Algorithm 2 for this example. The problem-specific implementation details are set out below.

As discussed in Subsection 8.1.1, we use the maneuver automaton representation with three trim trajectories. The local planning method (LPM) optimally connects two state-time points using an efficient geometric procedure.

The bound on the sampling region is determined by the maximum least connection cost J_B, which means that the sum of the cost of the optimal admissible path from the initial point to the sample and the cost of the optimal admissible path from the sample to a goal point should not exceed J_B. The initial value of J_B is chosen as 1.2 times the cost of the optimal admissible path between the initial and goal points. At each outer loop iteration of the algorithm, the multiplication factor is increased by 0.2 until J_B reaches twice the cost of the optimal admissible path between the initial and goal points, whereafter it is held constant. When a feasible path to goal has been found, the value of J_B is set to the cost of the feasible path – this ensures that the algorithm does not generate samples that could not possibly form part of a path with a lower cost than the feasible path already found. J_B is similarly adjusted when subsequent feasible paths with lower costs are found.

We use a four-dimensional sampling space: two dimensions to describe the airplane position, one dimension for the airplane velocity direction and one time dimension. For each value of J_B, we calculate the dimensions of a rectangular area that encloses the projection of the bounded sampling space on the plane depicting the airplane position, using the constant speed and cost function constraints. From this rectangular area, we generate a random sample of the airplane position from a uniform distribution. We also calculate the maximum time period allowed by the constant speed and cost function constraints and generate a random sample from a uniform distribution over this period. We then generate a random velocity direction sample from a uniform distribution. Next, we check if the position and time values of the generated sample lie within the cone in the position-time space as shown in Figure 7.5, where the shape of the cone is determined by the value of the constant nominal vehicle speed. We also check if the position and time values of the sample lie within the inverted cone centered at the goal position as shown in Figure 7.6. If the generated sample passes both these simple and efficient geometric tests, we generate the optimal admissible path from the initial point to the sample and the optimal admissible path from the sample to the goal point, using the LPM. If the sum of the costs of these paths is lower than J_B, the sample is added to the batch of samples, $\mathcal{Y}_c$, to be connected to the graph, $\mathcal{G}_T$.

We choose the size of the sample batch as $N_c = 20$ and the algorithm is restricted to 5 outer loop iterations. The conflict probability threshold for safe vehicle operation, P_C^{MAX}, is chosen as 0.05.

The motion planning algorithm was implemented in Python and the conflict detection algorithm implementation was taken from the simulation described in Section 5.1, which was written in Python with the bottleneck calculations written in C.

8.1.3 Results

A plot of the known reachable points and connecting paths for the airplane position dimensions for one outer loop iteration of the motion planning algorithm is shown in Figure 8.1. The mean and standard deviation of the results obtained from 100 simulations are shown in Table 8.1 where the algorithm is restricted to 5 outer loop iterations. For this problem setup, the cost of the optimal admissible path between the initial and goal points is 120. The sum of the mean and three times the standard deviation of the time to the first feasible path to the goal is less than 900 milliseconds, indicating that a feasible path to the goal can be generated in real-time. The mean cost of the first feasible path found is relatively low compared to the mean of the

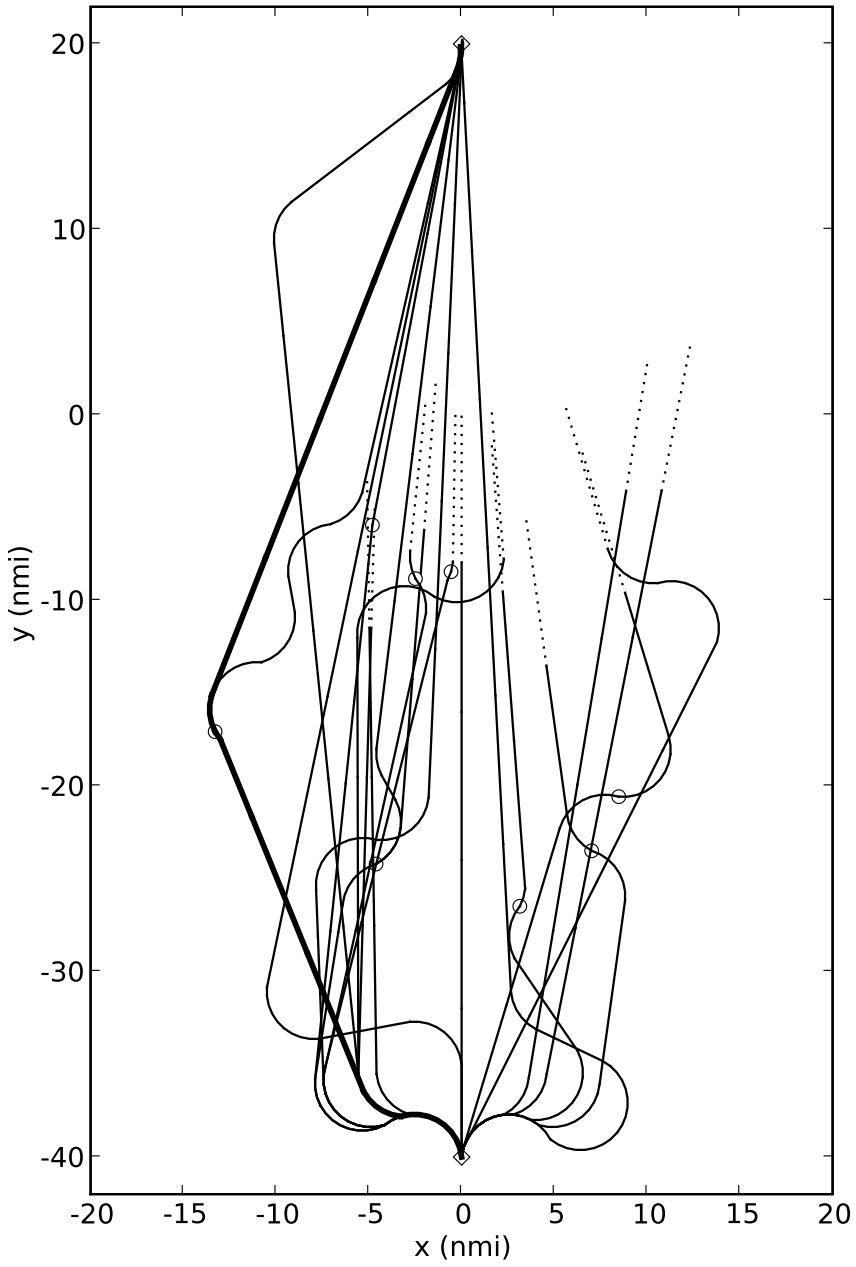


Figure 8.1: Known reachable points and connecting paths for one algorithm iteration (◇ – initial/goal point; o – known reachable point; best feasible path to goal shown with a thick line)

cost of the best path found when the algorithm has run for about 5 seconds.

To visualise the sampling space for this example problem, we plot the known reachable points and connecting paths for the position-time subspace of the sampling space in Figure 8.2. When comparing Figure 8.2 to Figure 7.6, it is evident that in the rejection sampling procedure, a large section of the hyperrectangle used for uniform sampling is rejected by the efficient geometric tests, indicating the value of using these preliminary tests.

We illustrate next the advantage of bounding the sampling region as described in Section 7.2 by presenting the results of two simulations with poorly selected sampling space bounds. For the first simulation, we employ a hyperrectangle for uniform sampling that is relatively large. We limit the allowed time to goal and apply the geometric rejection tests as described in

Table 8.1: Results of two airplanes example

First feasible path				Best path			
Time		Cost		Time		Cost	
Mean (s)	σ (s)	Mean	σ	Mean (s)	σ (s)	Mean	σ
0.589	0.097	140.4	7.1	4.911	0.686	128.9	1.3

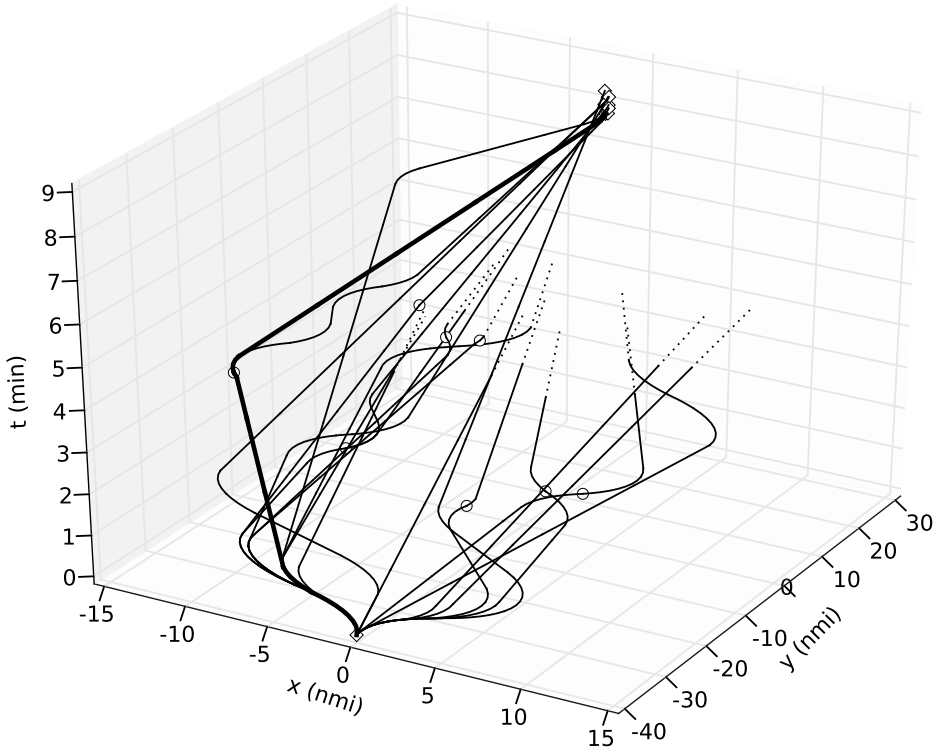


Figure 8.2: Known reachable points and connecting paths for one algorithm iteration, showing the position and time dimensions of the sampling space ($\diamond$ – initial/goal point; o – known reachable point; best feasible path to goal shown with a thick line)

Section 7.2, but place no further bounds on the sampling region. The results for this simulation are shown in Table 8.2. A comparison of Tables 8.1 and 8.2 shows that the execution times for

Table 8.2: Results of two airplanes example (large naive bounds on sampling region)

First feasible path				Best path			
Time		Cost		Time		Cost	
Mean (s)	σ (s)	Mean	σ	Mean (s)	σ (s)	Mean	σ
0.814	0.099	711.8	47.2	5.626	0.393	171.3	19.8

finding the first feasible path or execute 5 outer loop iterations do not differ significantly, but the quality (measured in cost) of the first feasible path and the best path found is significantly worse in the latter case. We did a similar simulation, but with the size of the hyperrectangle used for uniform sampling chosen quite small. The results of this simulation are shown in Table

8.3. A comparison of Tables 8.1 and 8.3 reveals that the execution times do not differ much,

Table 8.3: Results of two airplanes example (small naive bounds on sampling region)

First feasible path				Best path			
Time		Cost		Time		Cost	
Mean (s)	σ (s)	Mean	σ	Mean (s)	σ (s)	Mean	σ
0.572	0.065	225.5	33.6	4.138	0.209	140.3	3.2

but the quality of found paths is significantly worse in the latter case. Using the bounds on the sampling region as described in Section 7.2 therefore provides a beneficial manner in which to avoid the detrimental effects of choosing the sampling space badly.

Finally, we provide the results of another comparative simulation – this time generating a single sample for each outer loop iteration instead of generating a batch of samples for each iteration. This roughly corresponds to the method of Frazzoli et al.[25]. The means and standard deviations of the results of 100 simulations are shown in Table 8.4. Between Tables

Table 8.4: Results of two airplanes example (single point sampling)

First feasible path				Best path			
Time		Cost		Time		Cost	
Mean (s)	σ (s)	Mean	σ	Mean (s)	σ (s)	Mean	σ
0.111	0.085	140.0	6.2	4.846	0.742	129.1	1.2

8.1 and 8.4, similar results are visible, except for the mean time to the first feasible path, which is significantly shorter in the latter. The reason for this outcome is that the environment is relatively uncluttered. The motion planning algorithm of Chapter 7 only attempts to connect the goal point to the set of known reachable points after the connection of every point in the batch of samples to the graph $\mathcal{G}_T$ has been attempted. If an environment is such that a feasible path to goal is usually found after only a single batch of samples have been generated, it might be faster to use smaller batches of samples or even single sample points for each algorithm iteration.

From the results of the two-dimensional example in this section, it is clear that the motion planning algorithm proposed in Chapter 7 performs well for uncertain, dynamic and uncluttered environments and where the local planning method and the probabilistic conflict detection method are efficient. Despite the fact that the motion planning algorithm – including the LPM – is implemented in Python, which is a high-level interpreted language and is computationally not very efficient, it is possible to reliably generate a good quality feasible path to goal in real-time. In addition, the bounding of the sampling region using the maximum least connection cost bound prevents reduced quality feasible paths by preventing sampling region sizes that are chosen unwisely.

8.2 Autonomous Underwater Vehicle Example

In this section, we take the three-dimensional example problem from Section 5.2 – that of an AUV in a cluttered harbour environment – and apply the motion planning algorithm of Chapter 7 for conflict resolution.

8.2.1 Problem Description

For this example problem, we use the same vehicle and environment description as that used in Section 5.2: the boundary of the conflict volume is described by a triangulated surface and the uncertainty in the position and velocity of the AUV is described by a Gaussian distribution.

The conflict resolution problem for the current example consists of finding a feasible input that would steer the vehicle from the initial to the goal points. Differently from the example in Section 8.1, we employ the linear state space representation (Subsection 2.1.2) for the vehicle model. The input to this system is therefore a vector of continuous input variables.

We constrain the velocity of the vehicle by specifying a maximum speed limit. In addition, the time period within which the vehicle is required to reach the goal is limited to a maximum value. We also choose a minimum-input energy cost function – the best path to goal is therefore induced by the input with the lowest energy, but which still satisfies the maximum speed and time constraints.

8.2.2 Implementation

We implemented the motion planning algorithm of Chapter 7 as set out in pseudocode in Algorithm 2 for this example. The problem-specific details of the implementation is set out below.

The task of the local planning method (LPM) is to construct the optimal admissible input to steer the vehicle from one point to another. Because we use the state space representation for the vehicle model, the LPM has to assign values to a vector of continuous input variables for the given time period. In order to achieve this, we divide the allowed time period into smaller intervals and assume that the values of the variables in the input vector are held constant over each interval. We then employ a quadratic programming routine to assign values to these variables such that they yield an optimal input with respect to the minimum input energy cost function and satisfy the maximum speed inequality constraint as well as the time period and goal point equality constraint.

To adjust the value of the maximum least connection cost, J_B, we use the adaptive scheme described in Section 7.3. J_B is initialised to 2.0 times the cost of the optimal admissible path between the initial and goal states. The multiplication factor is increased by 1.0 after each batch of samples have been generated and tested for connection until a maximum multiplication factor of 5.0 is reached, after which it is held constant. When a feasible path to goal is found, the value of J_B is set to the cost of this path, ensuring that the algorithm does not attempt to connect samples that could not possibly yield a path with a lower cost than the feasible path already found. When feasible paths with lower associated costs are found, the value of J_B is similarly adjusted.

The chosen sampling space for this example is seven-dimensional: three dimensions to describe the vehicle position, three dimensions to describe the vehicle velocity, and one time dimension. For the rejection sampling procedure, we first randomly generate a sample from a uniform distribution on the seven-dimensional hyperrectangle encompassing the bounded sampling region. Most of the unsuitable samples are then rejected by using the efficient geometric test based on the maximum speed constraint and time limit as described in Section 7.2. The remaining samples are then tested by generating the optimal admissible path from the initial to the sample points and the optimal admissible path from the sample to the goal points, using the LPM – if the sum of the costs of these paths is lower than J_B, the sample is retained and added to the batch of samples to be tested for connection to the graph $\mathcal{G}_T$.

The number of samples per batch, N_c, is chosen as 10. We also implemented simulations with single sample generation per iteration as well as a batch of 40 samples per iteration. The results of these simulations are shown in Subsection 8.2.3. The chosen threshold for the conflict probability for safe operation is $P_C^{\mathrm{MAX}} = 0.05$.

The motion planning algorithm was implemented in Python with the conflict detection test implementation taken from the example simulation of Section 5.2, which was partially implemented in Python with the bottleneck calculations written in C.

8.2.3 Results

A plot of the known reachable points and connecting paths for one algorithm execution is shown in Figure 8.3. Differently from the example in Section 8.1, we attempt to find a feasible path and

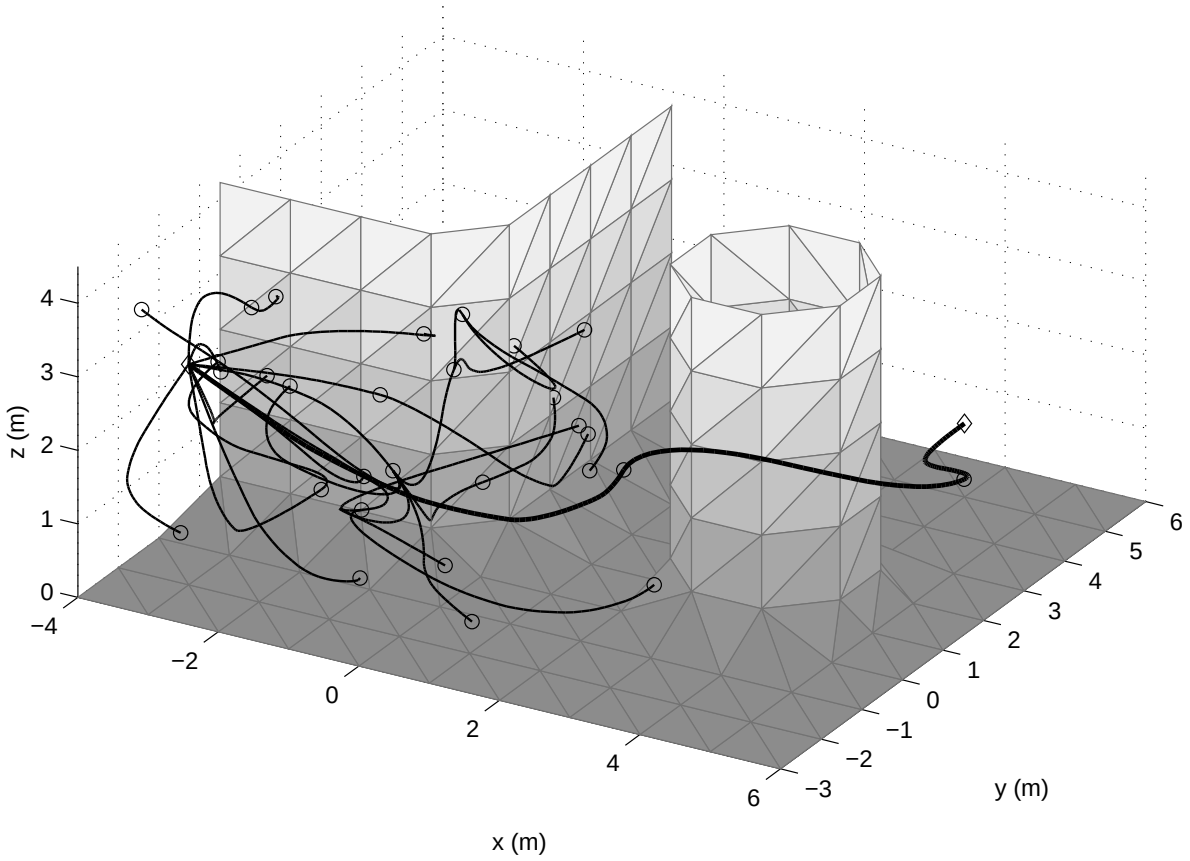


Figure 8.3: Known reachable points and connecting paths for one algorithm execution for the AUV example (⋄ – initial/goal point; o – known reachable point; feasible path to goal shown with a thick line)

search no further for better paths, due to the relatively long execution time of the algorithm. The means and standard deviations of the time to the first feasible path and the cost of the first feasible path for 100 simulations are shown in Table 8.5, where the cost of the optimal

Table 8.5: Results of AUV example (10 point sample batch)

First feasible path			
Time		Cost	
Mean (s)	σ (s)	Mean	σ
53.1	62.5	35.3	30.5

admissible path between the initial and goal states is 9.1. The sum of the mean and three times the standard deviation of the time to find a feasible path is about 240 seconds, therefore we

can only be fairly certain that the algorithm will find a feasible path within 4 minutes. This is clearly not fast enough for real-time computation, which is necessary for conflict resolution. One reason for the long execution time is the fact that the motion planning algorithm is implemented in Python, which is a high-level interpreted programming language and is not expected to be as efficient as low-level compiled languages like C. However, the conflict detection procedure was already partially implemented in C and an analysis of the algorithm time profile showed that the conflict detection procedure takes up 74% of the total execution time, indicating that not much time can be gained by implementing the motion planning algorithm in C. The algorithm spends about 19% of the total execution time in the LPM, indicating that it is also fairly inefficient.

The environment in this simulation is relatively cluttered, therefore we expect to see a performance increase when using batch sampling generation compared to single sampling point generation as discussed in Subsection 7.1.3. For the purpose of comparison, we provide the results of the motion planning algorithm adapted for single sample generation, shown in Table 8.6. When comparing Tables 8.5 and 8.6, it is evident that the results do not differ significantly,

Table 8.6: Results of AUV example (single point sample generation)

First feasible path			
Time		Cost	
Mean (s)	σ (s)	Mean	σ
51.8	54.3	35.3	47.6

with the single sample version yielding slightly better times than the first feasible paths, but yielding feasible paths with a slightly lower quality. We also provide the results of the motion planning algorithm adapted in order for a batch of 40 samples to be generated for each algorithm iteration as shown in Table 8.7. From a comparison between Tables 8.5 and 8.7, it is apparent

Table 8.7: Results of AUV example (40 point sample batch)

First feasible path			
Time		Cost	
Mean (s)	σ (s)	Mean	σ
49.4	44.4	43.3	71.1

that the latter version of the algorithm generates first feasible paths in a slightly shorter time, but it generates worse quality paths. From the comparative simulations, we conclude that there is no clear-cut advantage to using batch sample generation in cluttered environments.

The example simulation in this section shows the performance of the motion planning algorithm for a problem with a high-dimensional sampling space, inefficient LPM and relatively inefficient conflict detection procedure. Although it is not possible to generate a feasible path in real-time, the execution time of the algorithm is not excessive and the planner could provide a basis for subsequent research into more efficient methods.

Chapter 9

Conclusions

9.1 Summary

This thesis details the development of methods for conflict detection and resolution for an autonomous vehicle in an uncertain, dynamic and cluttered environment, where the combined vehicle and environment position and velocity states are modelled by a Gaussian probability distribution. The conflict region is allowed to be of a very general type: it may consist of multiple distinct sections, be an arbitrary shape, move or change shape. The vehicle and obstacle trajectories are also allowed to be complex as opposed to simple trajectories such as straight lines. The following paragraphs provide a brief summary of the methods and results contained in this thesis.

Chapter 1 explains conflict avoidance and proposes a unified architecture for the conflict avoidance system on a fully autonomous vehicle. When we study the architecture and the interfaces to the proposed probabilistic conflict detection method and motion planning algorithm for conflict resolution, it is evident that the interaction between the modules in the conflict avoidance system is stable.

Chapter 2 describes the two compatible representations for the vehicle model, namely the state space representation and the maneuver automaton representation. It also provides a summary of autonomous underwater vehicles and argues that they can be modelled using the state space or maneuver automaton representation, confirming that the methods developed for conflict detection and resolution in this thesis are applicable to such vehicles.

Part I describes the development of an efficient probabilistic conflict detection method using probability flow. Chapter 3 reviews conflict detection methods and presents a formal definition of the probabilistic conflict detection problem. Chapter 4 derives an expression for the probability of conflict using the flow of probability through the conflict region boundary for the general problem description. It then introduces the upper bound calculation and assumption of normally distributed vehicle states, both of which reduce the complexity of the problem. Lastly, it describes the adaptive integration procedure necessary for the real-time calculation of the integrals. The results of two simulation examples are expounded in Chapter 5 – these results show that the probability flow method compares well with existing methods for simple conflict regions and vehicle maneuvers and markedly outperforms the existing methods for complex and cluttered environments and complex[1] vehicle maneuvers. The second example shows that the probability flow method can calculate a tight upper bound to the probability of conflict in real-time for a general, three-dimensional environment.

[1] As opposed to simple trajectories such as straight lines.

Part II describes the adaptation of existing kinodynamic motion planning methods for conflict resolution in uncertain, dynamic and cluttered environments. In Chapter 6, existing conflict resolution methods are reviewed and important motion planning concepts introduced. Chapter 7 proposes some changes to the probabilistic roadmap methods for motion planning. These changes include using probabilistic conflict detection methods to handle uncertain environments, searching the state-time space instead of only the state space to handle dynamic environments and generating sample batches to handle cluttered environments. In addition, this chapter proposes an adaptive bound on the sampling space, namely the maximum least connection cost bound. Lastly, it combines the proposed changes and presents a motion planning algorithm for conflict resolution. The proposed motion planner is tested in Chapter 8, using the example problems of Chapter 5. The two-dimensional example shows that the motion planner can calculate a high quality and safe path in an uncluttered environment, in real-time. Comparative simulations without the bound on the sampling space show that the proposed adaptive bound prevents the generation of low quality paths when the sampling space size is chosen unwisely. The three-dimensional example shows that the motion planner has trouble generating a feasible path in real-time for cluttered environments. It also shows that the batch sample generation does not have a noticeable effect on the algorithm performance. For such environments, the motion planner is therefore not a complete solution, but rather a step in the right direction.

9.2 Primary Contributions

The following is a list of the most important contributions of the research presented in this thesis:

1. The development of the novel and efficient probabilistic conflict detection method is the most important contribution. Its advantage is twofold: firstly, it is able to manage a more general conflict region description and more complex vehicle trajectories than most existing methods, and secondly, it can calculate a tight upper bound to the probability of conflict in real-time, even for complex and cluttered environments and complex vehicle maneuvers. The conflict volume descriptions supported by this method include arbitrarily-shaped triangulated surfaces for the three-dimensional case, which is a very useful and general way to represent the environment. The only notable assumption of this method is that of Gaussian distributed position and velocity states. The Monte Carlo simulation method is the only existing method not using this assumption, but Monte Carlo methods are unable to execute in real-time for complex problem descriptions. We therefore conclude that the probability flow method presents a significant improvement in the field of conflict detection.

2. The motion planning algorithm adapted for conflict resolution in uncertain, dynamic and cluttered environments presents a novel method for conflict resolution. This algorithm contains two notable improvements: firstly, the way in which it manages uncertainty by using probabilistic conflict detection methods provides a robust solution compared to ad hoc and possibly unstable approaches of existing methods; secondly, the adaptive bound on the sampling space using the least connection cost metric prevents the generation of low quality feasible paths due to badly chosen bounds on the sampling space. The proposed motion planner can only generate feasible paths in real-time for simple environments, but it still presents a new approach in conflict resolution that could form the basis for subsequent improvements.

3. The unified architecture for conflict avoidance systems on fully autonomous vehicles enables robust conflict avoidance in uncertain, cluttered and dynamic environments. A vehicle path produced by the conflict resolution module always complies with the conflict

probability constraint because the conflict detection module is used in the design process of the vehicle path. The interaction between the conflict detection and conflict resolution modules will therefore always be without dissonance. The conflict resolution module also possesses the ability to rejoin the planned path. This means that the interaction between the path planner and the conflict resolution module will not contain any dissonance.

9.3 Future Work

We identify and briefly discuss the following promising avenues for future work:

1. The efficiency of calculating the probability flow for conflict detection could possibly be improved even further. This might be achieved by excluding from the numerical integration procedure those boundary sections that are located so far away from the vehicle position that the maximum probability flow through them is negligible.

2. The probability flow method for conflict detection could be extended to manage non-Gaussian vehicle and environment state distributions. If the joint position and velocity distribution of the vehicle relative to the conflict region in the transformed problem formulation (Subsection 3.2.2) can be described by a weighted sum of Gaussian probability density functions [3], it might still be possible to evaluate Equation 4.31 efficiently, which will allow efficient calculation of the probability of conflict for certain non-Gaussian distributions.

3. This thesis provides no guidelines to choose the conflict probability threshold for safe operation or the required accuracy of the conflict detection method. The effect that the choices of threshold and accuracy have on the conflict avoidance system performance should be investigated in order to set up guidelines for the specification and design of the conflict detection module.

4. There is much scope to improve the motion planning algorithm used for conflict resolution. For the probabilistic roadmap planners, putting more effort into placing samples might prove useful. Two such possible improvements include using deterministic sampling techniques and using non-uniform sample distributions.

5. The methods presented in this thesis are applicable to non-cooperative conflict avoidance. It might prove worthwhile to extend these methods to cooperative conflict avoidance. A first step in this direction would be to describe the dynamics of a conflict avoidance system using repeated replanning.

6. The modelling module of the conflict avoidance system was not extensively explored in this thesis. Although the interface between the modelling and other modules agree, the modelling module methods should be investigated and added to the simulation examples.

Appendix A

Adaptive Integration Error Analysis

A.1 Adaptive Integration using Simpson's rule

In the one-dimensional numerical integration procedure of Section 4.5 for the interval $[a, b]$ and integrand f, the value of the integral is computed as the sum of Simpson's rule evaluated over both halves of the interval, or

$$Q^S_{\left[a,\frac{1}{2}(a+b)\right]}f + Q^S_{\left[\frac{1}{2}(a+b),b\right]}f. \tag{A.1}$$

Using Equation 4.34, the expression of the difference between the actual integral and the estimated integral as given in Equation A.1 is given by

$$\begin{aligned}\varepsilon^S_{[a,b]} &= \int_a^b f(x)\,\mathrm{d}x - Q^S_{\left[a,\frac{1}{2}(a+b)\right]}f - Q^S_{\left[\frac{1}{2}(a+b),b\right]}f \\ &= -\frac{(b-a)^5}{2880}\left(\frac{1}{32}f^{(4)}(\xi_1) + \frac{1}{32}f^{(4)}(\xi_2)\right),\end{aligned} \tag{A.2}$$

for some $\xi_1 \in \left[a, \frac{1}{2}(a+b)\right]$ and some $\xi_2 \in \left[\frac{1}{2}(a+b), b\right]$. Using Equation 4.34, we can write the error rule as

$$\begin{aligned}E^S_{[a,b]}f &= \left|Q^S_{[a,b]}f - Q^S_{\left[a,\frac{1}{2}(a+b)\right]}f - Q^S_{\left[\frac{1}{2}(a+b),b\right]}f\right| \\ &= \frac{(b-a)^5}{2880}\left|f^{(4)}(\zeta) - \frac{1}{32}f^{(4)}(\xi_1) - \frac{1}{32}f^{(4)}(\xi_2)\right|\end{aligned} \tag{A.3}$$

for some $\xi_1 \in \left[a, \frac{1}{2}(a+b)\right]$, some $\xi_2 \in \left[\frac{1}{2}(a+b), b\right]$ and some $\zeta \in [a, b]$.

We can construct a simple bound on the error rule when we can find a bound on the size of the fourth derivative of the integrand, or

$$\left|f^{(4)}(\xi)\right| \leq f^{(4)}_{\max} \tag{A.4}$$

for the integration domain in question. The difference between the error rule and the size of the error in the estimated integral (as given in Equation A.2) is now calculated as

$$\begin{aligned}E^S_{[a,b]}f - \left|\varepsilon^S_{[a,b]}\right| = \frac{(b-a)^5}{2880}&\left|f^{(4)}(\zeta) - \frac{1}{32}f^{(4)}(\xi_1) - \frac{1}{32}f^{(4)}(\xi_2)\right| \\ &- \frac{(b-a)^5}{2880}\left|\frac{1}{32}f^{(4)}(\xi_1) + \frac{1}{32}f^{(4)}(\xi_2)\right|.\end{aligned} \tag{A.5}$$

When we apply the bound of Equation A.4, we obtain the following bound:

$$-\frac{(b-a)^5}{46080}f^{(4)}_{\max} \leq E^S_{[a,b]}f - \left|\varepsilon^S_{[a,b]}\right| \leq \frac{(b-a)^5}{2880}f^{(4)}_{\max}. \tag{A.6}$$

Equation A.6 indicates that the error rule can *underestimate* the error in the integral approximation by a much smaller margin than it can *overestimate* the integral approximation. The left-hand side of Equation A.6 can be used to specify the maximum integration interval length by bounding the possible error rule underestimation.

The bound on the fourth derivative of the integrand function as defined in Equation A.4 can in general be constructed by analysis of sample integrand functions. For the line integration of Equation 4.24, we can calculate the maximum fourth derivative of the integrand by assuming that the inner integral is constant, parametrising the position probability density function for a straight line and then calculating the fourth derivative of the integrand and finding its maximum absolute value.

Appendix B

Monte Carlo Simulation

In this appendix, we derive equations that have been used in the Monte Carlo simulations of the examples in Chapter 5. These simulations use the time-variant linear representation as set out in Subsection 2.1.2.

The Monte Carlo simulation generates L trajectories $\mathbf{x}_i(t)$ with given input $\mathbf{u}(t)$ and $\beta_i(t)$, which approximates a Wiener process with diffusion $\mathbf{Q}(t)$, such that Equation 2.7 holds for all $i \in [1, L]$, that is

$$\mathrm{d}\mathbf{x}_i(t) = \mathbf{A}(t)\mathbf{x}_i(t)\,\mathrm{d}t + \mathbf{B}(t)\mathbf{u}(t)\,\mathrm{d}t + \mathbf{B}_w(t)\,\mathrm{d}\beta_i(t) \tag{B.1}$$

The initial mean state value $\overline{\mathbf{X}}(t_0)$ and initial covariance $\mathbf{C}_{\mathbf{X}}(t_0, t_0)$ are specified. To generate appropriate initial values for the ith trajectory, the approach found in Peebles [64] and Ross [70]. For this approach, the initial covariance is decomposed using singular value decomposition,

$$\mathbf{C}_{\mathbf{X}}(t_0, t_0) = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \tag{B.2}$$

where $\mathbf{U}$ is an orthogonal matrix and $\mathbf{\Lambda}$ is a diagonal matrix with positive elements. A vector of independent Gaussian distributed random numbers are now generated with zero mean and variance according to the diagonal elements of $\mathbf{\Lambda}$. This vector is multiplied by $\mathbf{U}$ and added to $\overline{\mathbf{X}}(t_0)$ to yield an instance of a random vector with the correct mean and covariance.

The time period under consideration $[t_0, t_f]$ is now subdivided into intervals with length ΔT. To construct a trajectory, we assume that the values of the matrices $\mathbf{A}(t)$, $\mathbf{B}(t)$, $\mathbf{B}_w(t)$ and $\mathbf{C}_{\mathbf{X}}(t, t)$ are constant over one sampling period. The goal is to generate $\mathbf{\Phi}$, $\mathbf{\Gamma}$ and $\mathbf{\Gamma}_v$ such that the vehicle states at each sampling instant can be calculated according to

$$\mathbf{x}_i(t_k + \Delta T) = \mathbf{\Phi}(\Delta T)\mathbf{x}_i(t_k) + \mathbf{\Gamma}(\Delta T)\mathbf{u}(t_k) + \mathbf{\Gamma}_v(\Delta T)\mathbf{v}_i(t_k) \tag{B.3}$$

where $\mathbf{v}_i(t_k)$ is one vector of a set with $i \in [1, L]$ of which the distribution approximates a zero-mean Gaussian distribution with both the vector elements and vectors at different sampling instants independent from each other. The solution to Equation 2.9 is now given by Franklin et al. [22] as

$$\mathbf{\Phi}(\Delta T) = \mathbf{\Phi}(t, t + \Delta T) = e^{\mathbf{A}\Delta T} \tag{B.4}$$

This can also be written as

$$\mathbf{\Phi}(\Delta T) = \mathbf{I} + \mathbf{A}\Delta T\mathbf{\Psi}(\Delta T) \tag{B.5}$$

where

$$\mathbf{\Psi}(\Delta T) \triangleq \sum_{k=0}^{\infty} \frac{\mathbf{A}^k \Delta T^k}{(k+1)!} \tag{B.6}$$

Equations B.4 and B.5 can now be combined to produce

$$\mathbf{\Psi}(\Delta T) = \frac{1}{\Delta T}\mathbf{A}^{-1}\left(e^{\mathbf{A}\Delta T} - \mathbf{I}\right) \tag{B.7}$$

if $\mathbf{A}$ is invertible. $\mathbf{\Gamma}(\Delta T)$ is calculated as

$$\mathbf{\Gamma}(\Delta T) = \Delta T \mathbf{\Psi}(\Delta T)\mathbf{B} = \mathbf{A}^{-1}\left(e^{\mathbf{A}\Delta T} - \mathbf{I}\right)\mathbf{B} \tag{B.8}$$

The covariance matrix of $\mathbf{v}_i(t_k)$ as $L \to \infty$ is given by $\mathbf{Q}_d$ which is a diagonal matrix because of the independence of the vector elements. If the value of the state vector at t_k is known, the state covariance matrix at t_{k+1} is given by

$$\mathbf{C}_{\mathbf{X}}(t_{k+1}, t_{k+1}) = \mathbf{\Gamma}_v(\Delta T)\mathbf{Q}_d\mathbf{\Gamma}_v^T(\Delta T) \tag{B.9}$$

Also, when t_k is known, from the definition of covariance:

$$\mathbf{C}_{\mathbf{X}}(t_{k+1}, t_k) = \mathbf{0} \tag{B.10}$$

Equations 2.11, B.9 and B.10 combine to form

$$\mathbf{\Gamma}_v(\Delta T)\mathbf{Q}_d\mathbf{\Gamma}_v^T(\Delta T) = \int_0^{\Delta T} \mathbf{\Phi}(\xi)\mathbf{B}_w\mathbf{Q}\mathbf{B}_w^T\mathbf{\Phi}^T(\xi)\,\mathrm{d}\xi \tag{B.11}$$

The integral in Equation B.11 can be computed by using an approach similar to that of Van Loan [77]. According to this approach,

$$\exp\left(\begin{bmatrix} -\mathbf{A} & \mathbf{B}_w\mathbf{Q}\mathbf{B}_w^T \\ \mathbf{0} & \mathbf{A}^T \end{bmatrix}\Delta T\right) = \begin{bmatrix} \mathbf{F}_1(\Delta T) & \mathbf{G}_1(\Delta T) \\ \mathbf{0} & \mathbf{F}_2(\Delta T) \end{bmatrix} \tag{B.12}$$

where

$$\mathbf{G}_1(\Delta T) = e^{-\mathbf{A}\Delta T}\int_0^{\Delta T} e^{\mathbf{A}\xi}\mathbf{B}_w\mathbf{Q}\mathbf{B}_w^T e^{\mathbf{A}^T\xi}\,\mathrm{d}\xi \tag{B.13}$$

and

$$\mathbf{F}_2(\Delta T) = e^{\mathbf{A}^T\Delta T} \tag{B.14}$$

The integral in Equation B.11 is then given by

$$\int_0^{\Delta T} \mathbf{\Phi}(\xi)\mathbf{B}_w\mathbf{Q}\mathbf{B}_w^T\mathbf{\Phi}^T(\xi)\,\mathrm{d}\xi = \mathbf{F}_2^T(\Delta T)\mathbf{G}_1(\Delta T) \tag{B.15}$$

We now use singular value decomposition to obtain $\mathbf{\Gamma}_v(\Delta T)$ and $\mathbf{Q}_d(\Delta T)$ in Equation B.11 where $\mathbf{\Gamma}_v(\Delta T)$ is an orthogonal matrix and $\mathbf{Q}_d(\Delta T)$ is a diagonal matrix with positive elements. For each sampling instant, we generate a vector of Gaussian distributed independent random values of which the variances correspond to the diagonal elements of $\mathbf{Q}_d(\Delta T)$.

To estimate the probability of conflict, we simulate L trajectories and determine whether each trajectory enters the region of conflict. We then construct the set $E = \{E_1, E_2, \ldots, E_L\}$ with E_i set to 1 if $\mathbf{x}_i(t)$ entered the region of conflict and E_i set to 0 if $\mathbf{x}_i(t)$ did not enter the region of conflict for $t \in [t_0, t_f]$. As shown in Ross [70], the estimate of the probability of conflict is given by

$$P_C^{MC} = \frac{1}{L}\sum_{i=1}^{L} E_i \tag{B.16}$$

and the estimate of the variance is given by

$$\mathrm{Var}\left(P_C^{MC}\right) = \frac{1}{L}P_C^{MC}\left(1 - P_C^{MC}\right) \tag{B.17}$$

A fast implementation of the Monte Carlo simulation method for determining probability of conflict is given by Yang et al. [84].

Bibliography

[1] A. Pedro Aguiar, João P. Hespanha, and António M. Pascoal. Switched seesaw control for the stabilization of underactuated vehicles. *Automatica*, 43(12):1997–2008, 2007.

[2] A. Pedro Aguiar and António M. Pascoal. Dynamic positioning and way-point tracking of underactuated AUVs in the presence of ocean currents. *International Journal of Control*, 80(7):1092–1108, 2007.

[3] Daniel L. Alspach and Harold W. Sorenson. Nonlinear bayesian estimation using gaussian sum approximations. *IEEE Transactions on Automatic Control*, 17(4):439–448, 1972.

[4] Marco Attene and Michela Spagnuolo. Automatic surface reconstruction from point sets in space. *Computer Graphics Forum*, 19:457–465, 2000.

[5] Andrew Bacha, Cheryl Bauman, Ruel Faruque, Michael Fleming, Chris Terwelp, Charles Reinholtz, Dennis Hong, Al Wicks, Thomas Alberi, David Anderson, Stephen Cacciola, Patrick Currier, Aaron Dalton, Jesse Farmer, Jesse Hurdus, Shawn Kimmel, Peter King, Andrew Taylor, David Van Covern, and Mike Webster. Odin: Team VictorTango's entry in the DARPA Urban Challenge. *Journal of Field Robotics*, 25(8):467–492, 2008.

[6] Jérôme Barraquand, Bruno Langlois, and Jean-Claude Latombe. Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(2):224–241, 1992.

[7] Jérôme Barraquand and Jean-Claude Latombe. Robot motion planning: A distributed representation approach. *The International Journal of Robotics Research*, 10(6):628–649, 1991.

[8] Jérôme Barraquand and Jean-Claude Latombe. Controllability of mobile robots with kinematic constraints. Technical Report 61, Center for Integrated Facility Engineering, Stanford University, January 1992.

[9] Elizabeth Bone and Christopher Bolkcom. Unmanned aerial vehicles: Background and issues for congress. Technical report, Congressional Research Service, The Library of Congress, 2003.

[10] Valérie Boor, Mark H. Overmars, and A.Frank van der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *Proceedings of the IEEE International Conference on Robotics and Automation*, May 1999.

[11] John A. Borrie. *Stochastic Systems for Engineers.* Prentice Hall, 1992.

[12] M. Brookes. The matrix reference manual. [online] http://www.ee.ic.ac.uk/hp/staff/dmb/matrix/intro.html, 2005.

[13] Robert W. Button, John Kamp, Thomas B. Curtin, and James Dryden. *A Survey of Missions for Unmanned Undersea Vehicles.* RAND Corporation, 2009.

[14] Philip J. Davis and Philip Rabinowitz. *Methods of Numerical Integration*. Academic Press, 2nd edition, 1984.

[15] L.E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957.

[16] P. Encarnação and A. Pascoal. 3D path following for autonomous underwater vehicle. In *Proceedings of the 39th IEEE Conference on Decision and Control*, 2000.

[17] Hermann Engels. *Numerical Quadrature and Cubature*. Academic Press, 1980.

[18] Michael Erdmann and Tomás Lozano-Pérez. On multiple moving objects. *Algorithmica*, 2(4):477–521, 1987.

[19] Jonathan Evans, Pedro Patrón, Ben Smith, and David M. Lane. Design and evaluation of a reactive and deliberative collision avoidance and escape architecture for autonomous robots. *Autonomous Robots*, 24(3):247–266, 2008.

[20] Thor I. Fossen. *Guidance and Control of Ocean Vehicles*. John Wiley & Sons, 1994.

[21] T. Fraichard. Dynamic trajectory planning with dynamic constraints: A 'state-time space' approach. In *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems*, 1993.

[22] Gene F. Franklin, J. David Powell, and Michael Workman. *Digital Control of Dynamic Systems*. Addison Wesley Longman, 3rd edition, 1998.

[23] Emilio Frazzoli. *Robust Hybrid Control for Autonomous Vehicle Motion Planning*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 2001.

[24] Emilio Frazzoli, Munther A. Dahleh, and Eric Feron. Robust hybrid control for autonomous vehicle motion planning. In *Proceedings of the IEEE Conference on Decision and Control*, 2000.

[25] Emilio Frazzoli, Munther A. Dahleh, and Eric Feron. Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control, and Dynamics*, 25(1):116–129, 2002.

[26] Emilio Frazzoli, Munther A. Dahleh, and Eric Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics*, 21(6):1077–1091, 2005.

[27] Chiara Fulgenzi, Christopher Tay, Anne Spalanzani, and Christian Laugier. Probabilistic navigation in dynamic environment using rapidly-exploring random trees and gaussian processes. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, October 2008.

[28] Garratt Gallagher, Siddhartha Srinivasa, J. Andrew Bagnell, and David Ferguson. GATMO: a generalized approach to tracking movable objects. In *IEEE International Conference on Robotics and Automation*, May 2009.

[29] Roland Geraerts and Mark H. Overmars. A comparative study of probabilistic roadmap planners. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, December 2002.

[30] Michael J. Grimble and Michael A. Johnson. *Optimal Control and Stochastic Estimation: Theory and Applications*, volume 2. John Wiley & Sons, 1988.

[31] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 71–78, New York, USA, 1992. ACM.

[32] David Hsu, Lydia E. Kavraki, Jean-Claude Latombe, Rajeev Motwani, and Stephen Sorkin. On finding narrow passages with probabilistic roadmap planners. In *Proceedings of the third workshop on the algorithmic foundations of robotics on Robotics*, 1998.

[33] David Hsu, Robert Kindel, Jean-Claude Latombe, and Stephen Rock. Randomized kinodynamic motion planning with moving obstacles. *International Journal of Robotics Research*, 21(3):233–255, 2002.

[34] Inseok Hwang, Jesse Hwang, and Claire Tomlin. Flight-mode-based aircraft conflict detection using a residual-mean interacting multiple model algorithm. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2003.

[35] Léonard Jaillet, Anna Yershova, Steven M. LaValle, and Thierry Siméon. Adaptive tuning of the sampling domain for dynamic-domain RRTs. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, August 2005.

[36] Jonas Jansson and Fredrik Gustafsson. A framework and automotive application of collision avoidance decision making. *Automatica*, 44(9):2347–2351, 2008.

[37] Thomas Jones. *Real-Time Probabilistic Collision Avoidance for Autonomous Vehicles, Using Order Reductive Conflict Metrics.* PhD thesis, Massachusetts Institute of Technology, 2003.

[38] Thomas Jones. Tractable conflict risk accumulation in quadratic space for autonomous vehicles. *Journal of Guidance, Control, and Dynamics*, 29(1):39–48, 2006.

[39] Lydia E. Kavraki, Petr Švestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[40] Hye-Young Kim and Craig A. Woolsey. Global directional control of a slender autonomous underwater vehicle. *Journal of Guidance, Control, and Dynamics*, 30(1):255–259, 2007.

[41] M.J. Kochenderfer, J.P. Chryssanthacopoulos, L.P. Kaelbling, and T. Lozano-Perez. Model-based optimization of airborne collision avoidance logic. Technical Report ATC-360, Lincoln Laboratory, Massachusetts Institute of Technology, January 2010.

[42] James K. Kuchar. Methodology for alerting-system performance evaluation. *Journal of Guidance, Control, and Dynamics*, 19(2):438–444, 1996.

[43] James K. Kuchar and Lee C. Yang. Survey of conflict detection and resolution modeling methods. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 1997.

[44] James K. Kuchar and Lee C. Yang. A review of conflict detection and resolution modeling methods. *IEEE Transactions on Intelligent Transportation Systems*, 1(4):179–189, 2000.

[45] Prem K. Kythe and Michael R. Schäferkotter. *Handbook of Computational Methods for Integration.* Chapman & Hall/CRC, 2005.

[46] Jacoby Larson, Michael Bruch, and John Ebken. Autonomous navigation and obstacle avoidance for unmanned surface vehicles. *Proceedings of SPIE*, 6230(623007), 2006.

[47] Steven M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report 98–11, Computer Science Department, Iowa State University, October 1998.

[48] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[49] Steven M. LaValle, Michael S. Branicky, and Stephen R. Lindemann. On the relationship between classical grid search and probabilistic roadmaps. *International Journal of Robotics Research*, 23(7–8), 2004.

[50] Steven M. LaValle and James J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, 2001.

[51] Stephen R. Lindemann and Steven M. LaValle. Steps toward derandomizing RRTs. In *Proceedings of the Fourth International Workshop on Robot Motion and Control*, pages 271–277, June 2004.

[52] Pramod Maurya, E. Desa, A. Pascoal, E. Barros, G. Navelkar, R. Madhan, A. Mascarenhas, S. Prabhudesai, S. Afzulpurkar, A. Gouveia, S. Naroji, , and L. Sebastiao. Control of the MAYA AUV in the vertical and horizontal planes: Theory and practical results. In *Proceedings of the 7th IFAC Conference on Manoeuvring and Control of Marine Craft*, 2006.

[53] Emmanuel Mazer, Juan Manuel Ahuactzin, and Pierre Bessière. The Ariadne's Clew algorithm. *Journal of Artificial Intelligence Research*, 9:295–316, 1998.

[54] Mark H. Overmars. A random approach to motion planning. Technical Report RUU-CS-92-32, Utrecht University, October 1992.

[55] Russell A. Paielli and Heinz Erzberger. Conflict probability estimation for free flight. *Journal of Guidance, Control, and Dynamics*, 20(3):588–596, 1997.

[56] Russell A. Paielli and Heinz Erzberger. Conflict probability estimation generalized to non-level flight. *Air Traffic Control Quarterly*, 7(3), 1999.

[57] Russell P. Patera. General method for calculating satellite collision probability. *Journal of Guidance, Control, and Dynamics*, 24(4):716–721, 2001.

[58] Russell P. Patera. Satellite collision probability for nonlinear relative motion. *Journal of Guidance, Control, and Dynamics*, 26(5):728–733, 2003.

[59] Russell P. Patera. Calculating collision probability for arbitrary space-vehicle shapes via numerical quadrature. *Journal of Guidance, Control, and Dynamics*, 28(6):1326–1328, 2005.

[60] Russell P. Patera. Collision probability for larger bodies having nonlinear relative motion. *Journal of Guidance, Control, and Dynamics*, 29(6):1468–1471, 2006.

[61] Russell P. Patera. Space vehicle conflict-avoidance analysis. *Journal of Guidance, Control, and Dynamics*, 30(2):492–498, 2007.

[62] Russell P. Patera. Space vehicle conflict probability for ellipsoidal conflict volumes. *Journal of Guidance, Control, and Dynamics*, 30(6):1818–1821, 2007.

[63] Russell P. Patera and Glen E. Peterson. Space vehicle maneuver method to lower collision risk to an acceptable level. *Journal of Guidance, Control, and Dynamics*, 26(2):233–237, 2003.

[64] Peyton Z. Peebles, Jr. *Probability, Random Variables and Random Signal Principals.* McGraw-Hill, international edition, 1993.

[65] Per O. Pettersson and Patrick Doherty. Probabilistic roadmap based path planning for an autonomous unmanned aerial vehicle. In *ICAPS-04 Workshop on Connecting Planning Theory with Practice*, 2004.

[66] Maria Prandini, Jianghai Hu, John Lygeros, and Shankar Sastry. A probabilistic approach to aircraft conflict detection. *IEEE Transactions on Intelligent Transportation Systems*, 1(4):199–220, 2000.

[67] Maria Prandini, John Lygeros, Arnab Nilim, and Shankar Sastry. A probabilistic framework for aircraft conflict detection. In *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 1999.

[68] Maria Prandini, John Lygeros, Arnab Nilim, and Shankar Sastry. Randomized algorithms for probabilistic aircraft conflict detection. In *Proceedings of the IEEE Conference on Decision and Control*, pages 2444–2449, 1999.

[69] Filoktimon Repoulias and Evangelos Papadopoulos. Planar trajectory planning and tracking control design for underactuated AUVs. *Ocean Engineering*, 34:1650–1667, 2007.

[70] Sheldon M. Ross. *Simulation.* Academic Press, 3rd edition, 2002.

[71] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Pearson Education, second edition, 2003.

[72] Gildardo Sánchez and Jean-Claude Latombe. On delaying collision checking in PRM planning – application to multi-robot coordination. *International Journal of Robotics Research*, 21:5–26, 2002.

[73] Yuji Sato and Hiromitsu Ishii. Study of a collision-avoidance system for ships. *Control Engineering Practice*, 6(9):1141–1149, 1998.

[74] C. Silvestre and A. Pascoal. Depth control of the INFANTE AUV using gain-scheduled reduced order output feedback. *Control Engineering Practice*, 15(7):883–895, 2007.

[75] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, M. N. Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas M. Howard, Sascha Kolski, Alonzo Kelly, Maxim Likhachev, Matt McNaughton, Nick Miller, Kevin Peterson, Brian Pilnick, Raj Rajkumar, Paul Rybski, Bryan Salesky, Young-Woo Seo, Sanjiv Singh, Jarrod Snider, Anthony Stentz, William Whittaker, Ziv Wolkowicki, Jason Ziglar, Hong Bae, Thomas Brown, Daniel Demitrish, Bakhtiar Litkouhi, Jim Nickolaou, Varsha Sadekar, Wende Zhang, Joshua Struble, Michael Taylor, Michael Darms, and Dave Ferguson. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[76] Corné E. van Daalen and Thomas Jones. Fast conflict detection using probability flow. *Automatica*, 45(8):1903–1909, 2009.

[77] Charles F. van Loan. Computing integrals involving the matrix exponential. *IEEE Transactions on Automatic Control*, 23(3):395–404, 1978.

[78] Manuel Villén-Altamirano and José Villén-Altamirano. Analysis of restart simulation: Theoretical basis and sensitivity study. *European Transactions on Telecommunications*, 13(4):373–385, 2002.

[79] Chieh-Chih Wang, Charles Thorpe, and Sebastian Thrun. Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 842–849, 2003.

[80] Thomas Williamson and Ned A. Spencer. Development and operation of the Traffic Alert and Collision Avoidance System (TCAS). In *Proceedings of the IEEE*, pages 1735–1744, 1989.

[81] Lee C. Yang. *Aircraft Conflict Analysis and Real-time Conflict Probing using Probabilistic Trajectory Modeling*. PhD thesis, Massachusetts Institute of Technology, 2000.

[82] Lee C. Yang and James K. Kuchar. Prototype conflict alerting system for free flight. *Journal of Guidance, Control, and Dynamics*, 20(4):768–773, 1997.

[83] Lee C. Yang and James K. Kuchar. Using intent information in probabilistic conflict analysis. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 1998.

[84] Lee C. Yang, Ji H. Yang, James K. Kuchar, and Eric Feron. A real-time Monte Carlo implementation for computing probability of conflict. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, pages 617–631, 2004.

[85] Yuandong Yang and Oliver Brock. Adapting the sampling distribution in PRM planners based on an approximated medial axis. In *Proceedings of the IEEE International Conference on Robotics and Automation*, April–May 2004.

[86] Anna Yershovay, Léonard Jaillet, Thierry Siméonz, and Steven M. LaValle. Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain. In *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2005.

[87] J. Yuh. Design and control of autonomous underwater robots: A survey. *Autonomous Robots*, 8(1):7–24, 2000.